150 ptas.
42
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IV - Fascículo 42

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-005-8 (tomo 4)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 318410
Impreso en España - Printed in Spain - Octubre 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





Útil compañía

La última generación de ordenadores portátiles convierte a su pionero, el Osborne, en una máquina incómoda y pesada

La definición de "portátil" se ha tenido que revisar desde que apareciera la última generación de ordenadores "de mano". En realidad, a los micros portátiles que se introdujeron hace unos pocos años ahora se los denomina "transportables". Los ordenadores que en la actualidad ofrecen una genuina portabilidad son los que llevan su propia fuente de alimentación eléctrica, visualización y dispositivos de almacenamiento en un paquete no más grande que una guía telefónica.

El Epson HX-20 fue uno de los primeros que ofrecieron este tipo de portabilidad, pero ahora su diminuta visualización en cristal líquido de 20 caracteres por 4 líneas, refleja la edad de la máquina. Los últimos portátiles, como el Tandy 100, el NEC PC-8201A y el Olivetti M10 tienen todos un precio similar pero pueden visualizar en sus pantallas cuatro veces más caracteres.

¿Y qué pueden hacer estos ordenadores? ¿Cuáles son sus ventajas y sus desventajas respecto de los micros convencionales de sobremesa? La razón más obvia para adquirir un portátil es la de disponer de un potencial informático completo en cualquier lugar y en cualquier momento. Muchas personas pasan gran parte de su tiempo alejadas del ordenador de su escritorio, y muchas horas improductivas transcurren en otras oficinas, habitaciones de hotel, aeropuertos y trenes. El ordenador portátil permite aprovechar este tiempo muerto.

La más reciente generación de portátiles proporciona un adecuado potencial informático personal para trabajos científicos y de ingeniería, contabilidad, administración financiera y tratamiento de textos; de hecho, prácticamente para cualquier aplicación para la que se utilicen los ordenadores personales convencionales.

Los ordenadores de mano suelen llevar al menos tres programas incorporados: un intérprete de BASIC, un programa para tratamiento de textos y software para comunicaciones. El Tandy 100 y el Olivetti también están equipados con programas de planificación y agenda de direcciones; así, el usuario puede disponer de señas, números de teléfono y las entrevistas previstas para cada día.

El programa para comunicaciones es sumamente importante porque permite que el portátil se ponga en contacto con otros micros y bases de datos a través de la red de teléfonos. Esta facilidad también puede convertir el micro portátil en un terminal de télex o receptor y transmisor de correo electrónico. Por supuesto, para conseguirlo se ha de emplear un modem o un acoplador acústico. De esta forma, un ejecutivo que no se halle en su despacho se puede mantener en contacto con su oficina central. Un periodista puede escribir su artículo *in situ* y transmitirlo inmediatamente al ordenador del periódico.

Los ordenadores portátiles más caros, como el Sharp PC-5000 y el Epson PX-8, utilizan los sistemas operativos MS-DOS y CP/M, que son comunes a sus equivalentes de sobremesa. Son, por consiguiente, capaces de ejecutar una amplia gama de software de oficina.



Tony Sleep

Sobre la marcha
La informática sobre la marcha se está haciendo cada vez más usual, principalmente entre los hombres de negocios. Algunos están utilizando la nueva generación de ordenadores "de mano" para ganar unos pocos minutos y emplearlos en tratamiento de textos durante las esperas en salas y aeropuertos antes de ir de un taxi a un tren y de éste al avión. Otros, como los vendedores, están abriendo un nuevo camino llevándose el ordenador consigo en sus visitas a clientes, pudiendo generar al instante presupuestos cuya preparación, de otra forma, ocuparía varios días. Los ejecutivos pueden, sobre la marcha, enviar datos a la oficina central utilizando un modem y las líneas telefónicas normales, o, al final del día, regresar a la oficina y enviar los datos directamente a un ordenador más grande

El Epson PX-8 viene con el popular paquete de tratamiento de textos Wordstar ya instalado en sus chips de ROM. El Sharp utiliza cartuchos conectables de memoria de burbujas que proporcionan 128 Kbytes de almacenamiento extra cada uno. Estos cartuchos tratan los datos a una velocidad mucho mayor que las unidades de disco.

Los portátiles menos caros utilizan programas para aplicaciones concretas que suelen cargarse en la RAM del ordenador desde cassette. Éste es un proceso mucho más lento que cargar desde cartucho de memoria de burbujas o desde unidad de disco. El NEC PC-8201A viene con una cassette que contiene varios programas de aplicaciones.

Informática a bordo
Todo hombre de negocios que desee utilizar su ordenador portátil durante un viaje en avión tendrá que poner especial atención al elegir la compañía aérea. Oficialmente, las autoridades de aviación civil afirman que los equipos electrónicos y a pilas pueden producir interferencias en los controles de vuelo de los aviones. Pero cada compañía interpreta estos informes de forma distinta. La línea alemana Lufthansa y la australiana Qantas prohíben todo equipo eléctrico a bordo. La norteamericana Pan Am no permite radios ni grabadoras, pero sí autoriza juegos electrónicos y ordenadores. Japan Airlines no desautoriza equipos eléctricos de ninguna clase, a diferencia de las líneas británicas, que prohíben los equipos electrónicos

Entre ellos se incluyen una calculadora con memoria, un formateador de textos, un gestor de cartera de inversiones y un evaluador de préstamos. La calculadora con memoria convierte a la máquina en una calculadora que puede recordar hasta 99 entradas. El formateador de textos prepara para la impresión archivos entrados por el programa para tratamiento de textos, pudiendo especificar anchura de márgenes, división del texto en páginas, asignación de números de página, etc. El gestor de cartera de inversiones es para que lo empleen las personas que desean evaluar cómo marchan sus acciones y sus participaciones. Este programa analiza una cartera de hasta 50 inversiones, calculando pérdidas y beneficios.

Al igual que cualquier otro ordenador, el micro portátil se puede conectar a periféricos como impresoras, grabadoras de cassette y modems. Aparte del factor obvio de tamaño y peso, la prueba de fuego de un auténtico portátil es que ha de funcionar a pilas, tener su propia visualización y transportar consigo en ROM sus programas para tratamiento de textos y comunicaciones.

Hay máquinas, como el Apple IIc y el Apricot, que se anuncian como portátiles. Pero éstas no se pueden utilizar en tránsito, ya que se tienen que enchufar a una toma de corriente, conectar a una pantalla y cargar sus programas en RAM desde disco. Prescindiendo de su tamaño más pequeño y de su menor peso, estas máquinas tienen más cosas en común con el micro de sobremesa que con el ordenador de mano a pilas.

Además de sus pilas principales, los ordenadores portátiles están equipados con pequeñas pilas de níquel-cadmio recargable que pueden proporcionar energía en casos de emergencia. Esto es esencial, ya que de no contar con este apoyo se perderían todos los datos si se produjera un fallo de las pilas principales.

La mayoría de los portátiles también poseen una interface para lector de códigos de barras, de modo que se los puede utilizar para el control de existencias. El lector de códigos de barras se pasa por encima del código de barras impreso en los paquetes de productos. Este decodifica la información relativa a precio y fecha, que puede ser procesada por el ordenador para proporcionarles a los comerciantes una lectura exacta del inventario de su stock. De las máquinas que hemos ilustrado en estas páginas, el Tandy Modelo 100, el NEC PC-8201A y el Olivetti M10 vienen equipados con lector de códigos de barras, pero el Casio FP 200 no lo posee. Las tres primeras máquinas en realidad son muy similares en numerosos aspectos, porque todas están hechas en la misma fábrica japonesa. Las diferencias significativas entre ellas son que la Olivetti tiene una pantalla inclinable, la NEC tiene menos software incorporado y las memorias del Tandy y del Olivetti no se pueden ampliar a más de 32 Kbytes, mientras que el NEC se puede ampliar hasta 64 Kbytes. Asimismo, el NEC utiliza cartuchos de memoria de 32 Kbytes intercambiables que retienen sus datos aun cuando se desconecten del ordenador. Todas las mejoras de memoria para ordenadores manua-



Epson HX-20
Aunque posee una pantalla pequeña, el HX-20 tiene la ventaja de llevar incorporadas una grabadora de cassette y una diminuta impresora. También incluye un modesto procesador de textos

Casio FP-200
El Casio es uno de los ordenadores de mano más baratos, pero carece de un procesador de textos incorporado. En cambio, ofrece una versión de hoja electrónica

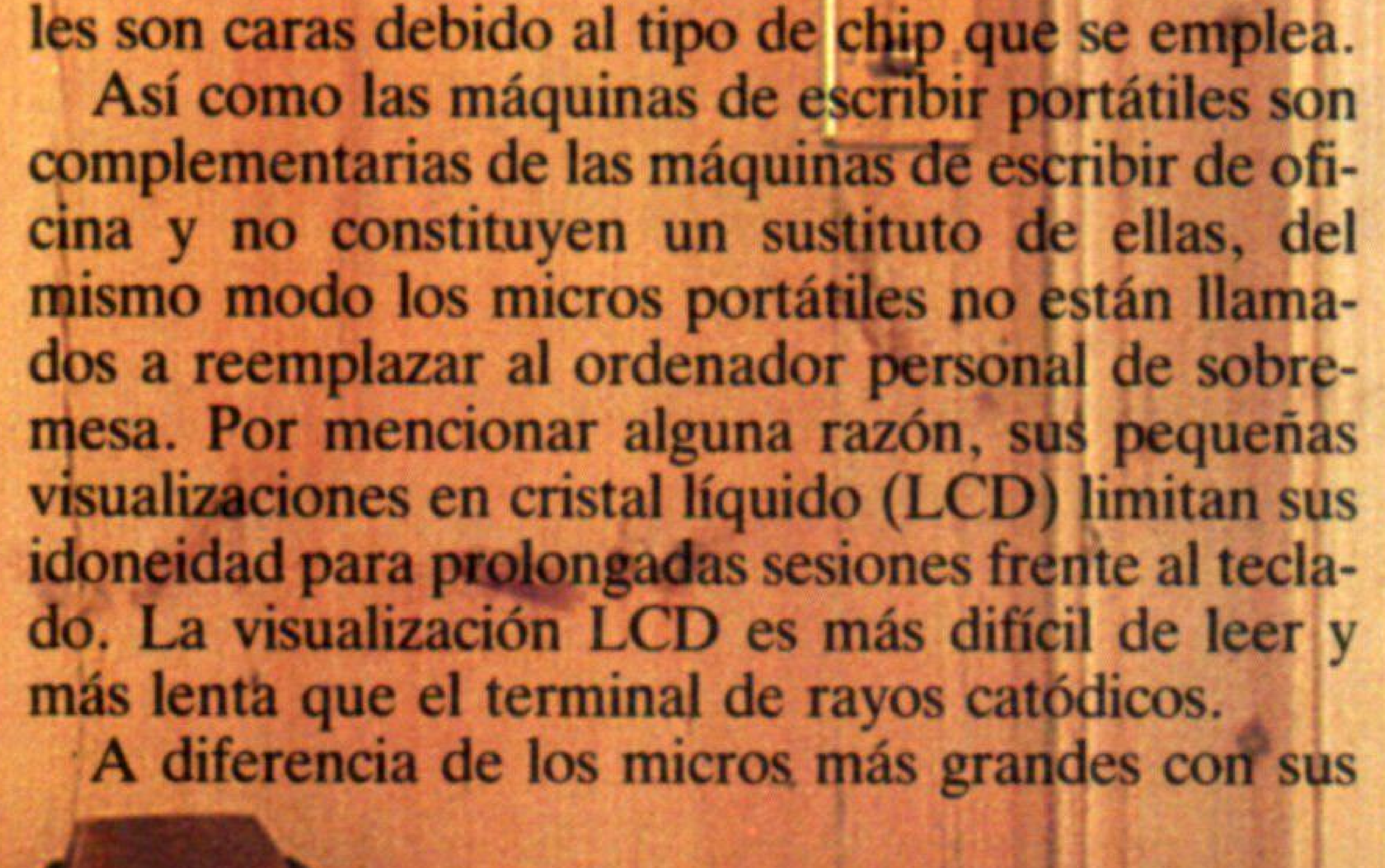


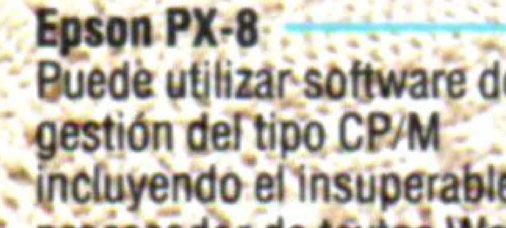
les son caras debido al tipo de chip que se emplea.

Así como las máquinas de escribir portátiles son complementarias de las máquinas de escribir de oficina y no constituyen un sustituto de ellas, del mismo modo los micros portátiles no están llamados a reemplazar al ordenador personal de sobremesa. Por mencionar alguna razón, sus pequeñas visualizaciones en cristal líquido (LCD) limitan sus idoneidad para prolongadas sesiones frente al teclado. La visualización LCD es más difícil de leer y más lenta que el terminal de rayos catódicos.

A diferencia de los micros más grandes con sus teclados inclinados, los ordenadores portátiles más económicos poseen teclados planos cuya utilización es incómoda para el usuario. Además los portátiles del extremo inferior del mercado no pueden ejecutar los programas de oficina basados en disco.

Pero, con todo, el ordenador portátil de mano está aquí y su permanencia está asegurada. La difusión del empleo de micros les está mostrando a muchas personas cómo el potencial del ordenador las puede ayudar a llevar sus vidas con mayor eficacia. El portátil les permite acceder a este potencial estén donde estén.





Epson PX-8
Puede utilizar software de gestión del tipo CP/M incluyendo el insuperable procesador de textos Wordstar, que viene con la máquina



Tandy TRS-80 Modelo 100/NEC PC-8201A/Olivetti M10
Constituyen distintas versiones reempaquetadas del mismo micro. Comparten un excelente procesador de textos incorporado y una buena gama de interfaces. En la fotografía también vemos un modem a pilas de Olivetti

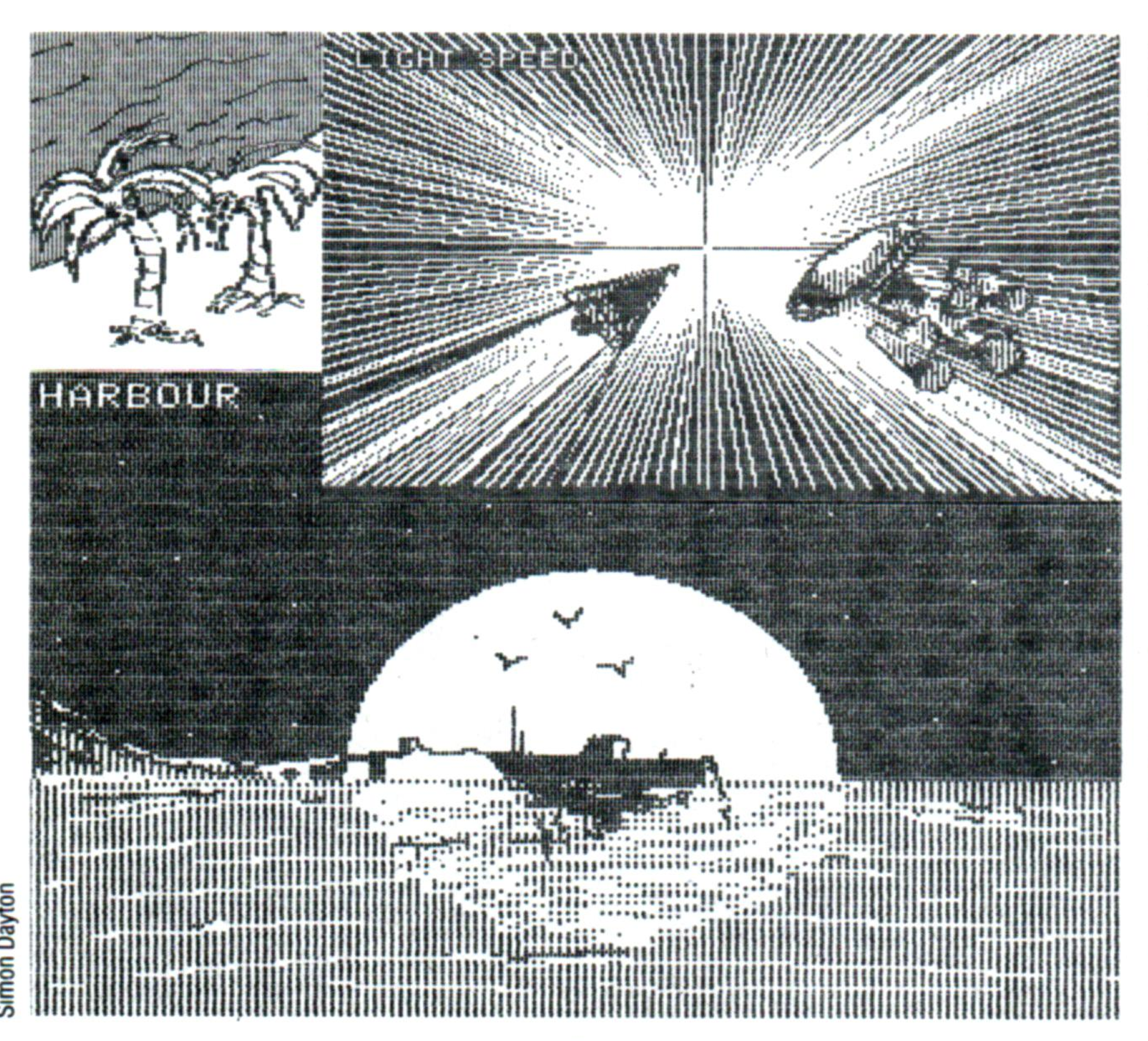
Marcus Wilson-Smith

Modelo	Memoria estándar	Memoria máxima	Pantalla	Peso
Casio FP-200	8 K	32 K	8 × 20	1,4 kg
Epson HX-20	16 K	32 K	4 × 20	1,8 kg
Epson PX-8	64 K	64 K + 120 K*	8 × 80	2,3 kg
NEC PC-8201A	16 K	64 K + 32 K*	8 × 40	1,8 kg
Olivetti M10	8 K	32 K	8 × 40	1,8 kg
Tandy TRS-80 Modelo 100	8 K	32 K	8 × 40	1,8 kg

* El NEC puede usar un cartucho de RAM de 32 K y el Epson PX-8 un disco de RAM de 120 K.

Primeras impresiones

En este capítulo mostramos cómo preparar una impresora para producir atractivos gráficos y cómo crear un programa de "vuelco de pantalla"



Simon Dayton

Impresión por pantalla
Estos diseños se dibujaron en pantalla utilizando una pastilla para gráficos. El contenido de las pantallas se volcó luego a una impresora Epson FX-80, mostrando las posibilidades para gráficos de una impresora matricial

La mayoría de los ordenadores personales poseen una modalidad de gráficos en baja resolución en que las imágenes se construyen en la pantalla a partir de caracteres para gráficos, cada uno de éstos de igual tamaño que un carácter para texto convencional. Estos caracteres de "bloques" poseen códigos de caracteres superiores a 127, ya que los números del 0 al 127 están reservados para el juego de caracteres ASCII. De modo que PRINT CHR$(90) imprimirá en la pantalla un carácter ASCII ("Z", en este caso), mientras que PRINT CHR$(128) visualiza un carácter para gráficos (o un rectángulo negro si se está utilizando un micro Dragon).

Para imprimir la letra "Z" en una impresora digitaríamos LPRINT CHR$(90), por lo que se podría pensar que, del mismo modo, LPRINT CHR$(128) imprimirá en el papel un rectángulo negro. Pero, lamentablemente, no es éste el caso. Ello se debe a que los caracteres por encima del código 127 varían muchísimo de una marca a otra de micro y, obviamente, los fabricantes de impresoras no pueden producir una impresora especial para cada ordenador que haya en el mercado. Lo que los fabricantes tienden a hacer es o bien copiar el juego ASCII estándar en los códigos del 128 al 255 o, alternativamente, programar sus propios caracteres.

La gama de impresoras Epson no viene con ningún carácter para gráficos. En cambio, el usuario puede cambiar cualquiera de los caracteres ASCII estándar para producir sus propios caracteres para gráficos. Esto se consigue enviándole a la impresora "códigos de escape" (véanse pp. 804 y 805).

Los gráficos de ordenador en alta resolución se construyen a partir de pequeños puntos o pixels y no a partir de caracteres enteros. De un modo similar, la impresión en alta resolución utiliza pequeños puntos de tinta. El cabezal de impresión de una impresora matricial posee cierta cantidad de agujas dispuestas en una línea vertical que se desplaza a través del papel a medida que va imprimiendo. Por lo general, los caracteres se componen en una matriz de puntos (de 8 × 8 puntos, p. ej.). No obstante, se pueden producir gráficos controlando las agujas individualmente.

El primer paso consiste en colocar la impresora en modalidad para gráficos. Al igual que si se tratara de cualquier otro ejercicio de impresión, esto se realiza enviándole un código de escape que es específico para el tipo de impresora que se está utilizando. En la Epson FX-80, por ejemplo, las instrucciones necesarias son:

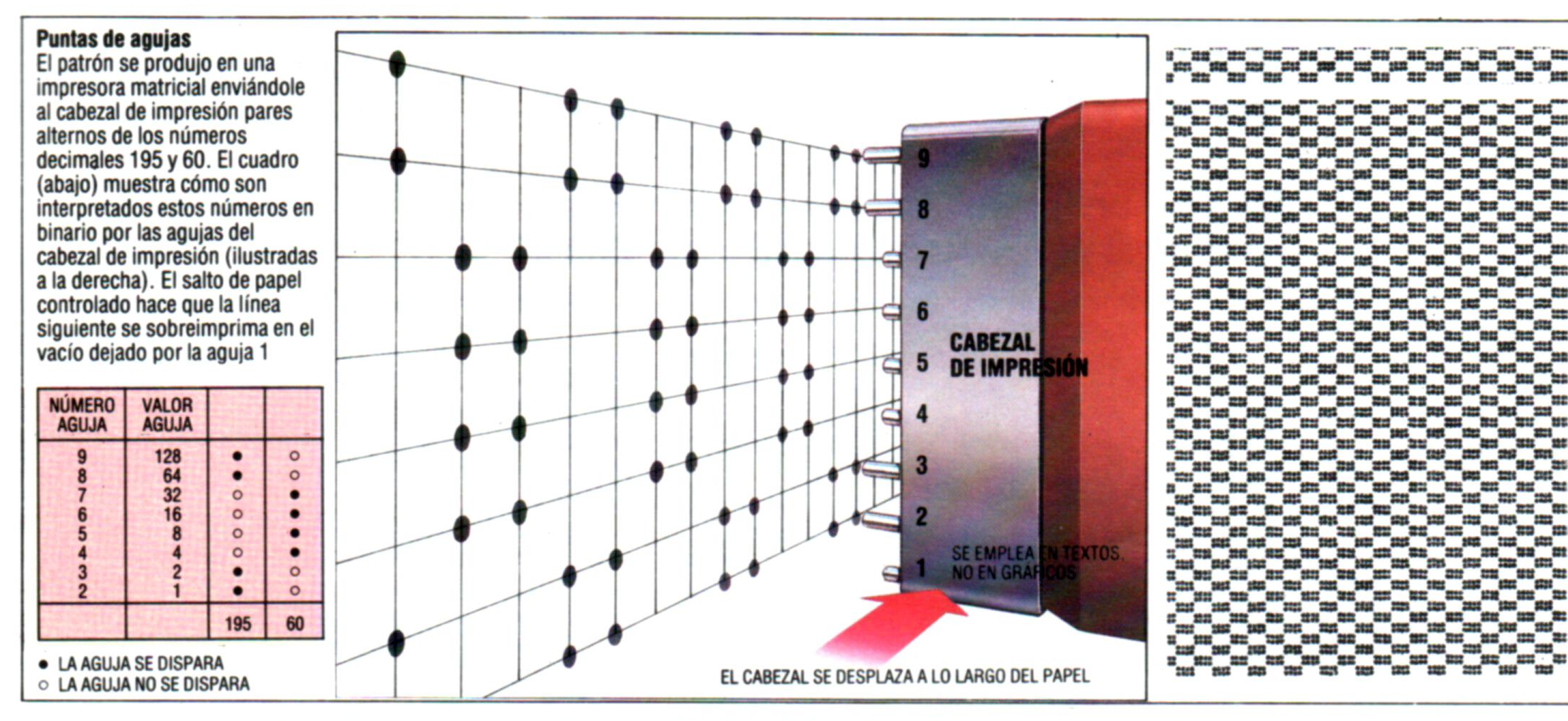
```
LPRINT CHR$(27);"K";CHR$(N1);(N2);
```

La letra "K" indica modalidad de gráficos y los numeros (N1) y (N2) establecen la anchura de cada línea de gráfico; en otras palabras, el número de puntos que entrarán a lo ancho de la página.

Estando en modalidad de gráficos estándar, la FX-80 puede imprimir un máximo de 480 puntos en una línea. Otras modalidades permiten resoluciones desde 576 a 1 920 puntos por línea. Por lo tanto, si deseamos utilizar la anchura total, la longitud de línea requerida será 480. En nuestro código son necesarios dos números para establecer la anchura porque el tamaño máximo de cada número es 255. El segundo número (N2) se multiplica, por consiguiente, por 256 y se le suma al primero, (N1). De modo que para 480 los números son 1 y 224 ($480 = 256 \times 1 + 224$). En consecuencia, en la Epson FX-80 se requiere la siguiente instrucción:

```
LPRINT CHR$(27);"K";CHR$(224);CHR$(1);
```

Habiendo programado la impresora con la longitud de línea de gráficos, sólo falta enviar los datos del gráfico. Aun cuando en el cabezal de impresión de una Epson FX-80 hay nueve agujas, en la mayoría de las modalidades para gráficos únicamente se pueden emplear las ocho superiores. Empezando desde la patilla inferior, las numeramos 1, 2, 4, 8, 16, 32, 64 y 128. Los datos para las ocho agujas se pueden representar, entonces, mediante un único número, entre 0 y 255, y éste se le envía a la impresora utilizando LPRINT CHR$(X), donde X es el número. De modo que si sólo deseáramos "disparar" la aguja inferior, enviaríamos CHR$(1) a la impreso-



Puntas de agujas
El patrón se produjo en una impresora matricial enviándole al cabezal de impresión pares alternos de los números decimales 195 y 60. El cuadro (abajo) muestra cómo son interpretados estos números en binario por las agujas del cabezal de impresión (ilustradas a la derecha). El salto de papel controlado hace que la línea siguiente se sobreimprima en el vacío dejado por la aguja 1

NÚMERO AGUJA	VALOR AGUJA		
9	128	●	○
8	64	●	○
7	32	○	●
6	16	○	●
5	8	○	●
4	4	○	●
3	2	●	○
2	1	●	○
		195	60

● LA AGUJA SE DISPARA
○ LA AGUJA NO SE DISPARA

ra; para activar sólo la aguja superior enviaríamos CHR$(128). Para una combinación de patillas simplemente vamos sumando los números de cada aguja. Este proceso se repite luego para cada una de las 480 posiciones de pixel de la página.

En la ilustración hay dos patrones de agujas: CHR$(195) y CHR$(60). Así, para imprimir las cuatro primeras columnas del patrón de línea se digita:

```
LPRINT CHR$(195);CHR$(195);CHR$(60);CHR$(60);
```

Después de cuatro columnas el patrón se repite, por lo cual un bucle FOR...NEXT se ocupa del resto de la línea.

Es importante comprender que, en el ejemplo, CHR$(60) no indica a la impresora que imprima el carácter ASCII con código 60: es una manera de representar los datos para las agujas del cabezal de impresión. La impresora lo reconoce como tal porque previamente hemos transmitido la secuencia CHR$(27);"K" para activar la modalidad de gráficos.

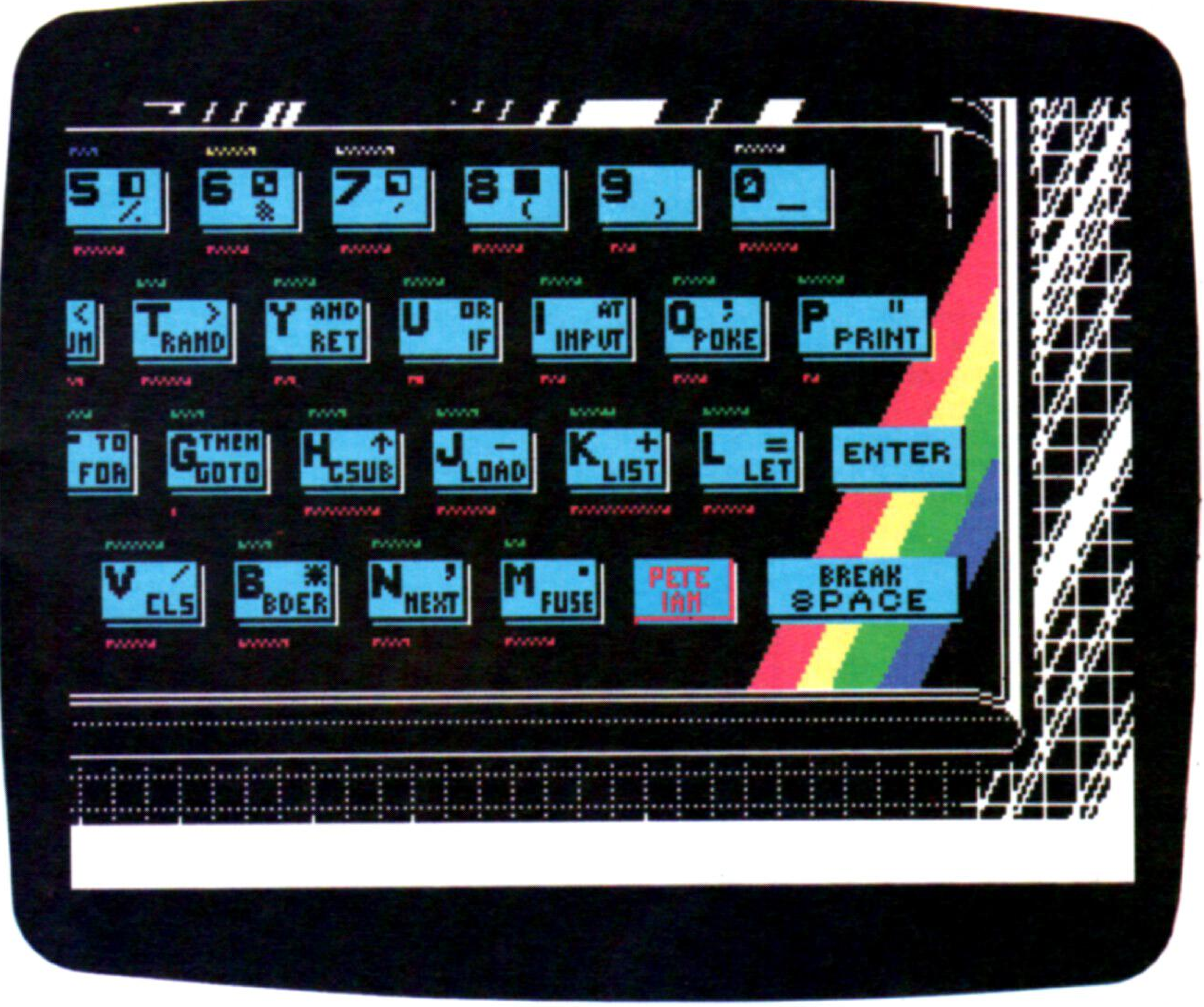
Este método de impresión, que se conoce como *impresión de imagen de bits*, se describe aquí para una impresora Epson FX-80; otras impresoras utilizan un procedimiento similar, pero los detalles exactos pueden variar. Producir gráficos de esta forma es bastante laborioso y sólo es realmente adecuado para patrones. Una forma mucho mejor de imprimir gráficos es mediante un *vuelco de pantalla*. Éste es un programa que copia en el papel lo que está visualizado en la pantalla.

Explorando a lo ancho y a lo alto de la visualización en pantalla, el programa verifica si el pixel está encendido en cada posición. Si lo está, entonces es necesario que una aguja del cabezal de impresión se dispare en la posición correspondiente del papel. La exploración se realiza utilizando la función POINT(x,y) o instrucciones similares disponibles en la mayoría de micros; si un pixel está iluminado, la función es 1; si no lo está, ésta corresponderá a 0. Las distintas resoluciones de los diferentes micros implican la realización de algunos ajustes.

Una pregunta que quizá se le ocurra es: ¿cómo manipula un programa de vuelco de pantalla las visualizaciones en color? La solución habitual consiste en utilizar distintos patrones de puntos para cada color. Un pixel de pantalla que fuera negro se podría imprimir aplicando cuatro puntos en forma de cuadrado; uno que fuera rojo se podría representar mediante dos puntos, y uno que fuera blanco no utilizaría ningún punto. La función POINT(x,y) produce un número diferente según el color del pixel.

Los programas para vuelco de pantalla se suelen añadir en forma de subrutinas al final del programa que produce la imagen. Para "volcar" la imagen a la impresora se podría pulsar la tecla "P": el programa saltaría, entonces, a la subrutina. Un programa para vuelco de pantalla escrito en BASIC tiende a ser bastante lento, tardando alrededor de cinco minutos en imprimir una pequeña imagen. Las versiones en lenguaje máquina son ligeramente más rápidas.

Como puede verse, las capacidades para gráficos de las impresoras matriciales son avanzadas, aunque un poco incómodas de utilizar. Sin embargo, una vez dominadas, la página impresa puede ser tan atractiva como la visualización en pantalla.



Salpicones de color
Esta imagen del teclado del Spectrum se produjo en una impresora de chorro de tinta en color; en realidad se trata de una impresora matricial en la cual las agujas se han sustituido por chorros de tinta

Batalla naval

He aquí un interesante programa de juegos para la red más asequible para el usuario: Sinclair ZX Net

La batalla naval es un juego clásico que se acostumbra jugar con lápiz y papel. Cada jugador posee dos cuadrículas, que mantiene ocultas a su adversario. En una de ellas tiene señalados sus propios barcos, en la otra va marcando sus resultados a medida que "dispara" contra los barcos de su oponente; es decir, se intenta adivinar la posición de éstos.

El programa que hemos desarrollado funciona con dos Sinclair Spectrum conectados mediante una red. Ambas máquinas han de estar equipadas con la Interface 1. Cada jugador se sienta frente a su propia pantalla, y los dos ordenadores se envían mensajes que informan hacia dónde están disparando los jugadores y qué resultados obtienen.

La primera dificultad que se encuentra es el identificar a los jugadores. Con el fin de comunicarse, cada Spectrum posee un número de estación de red diferente. Los dos ordenadores empiezan con programas idénticos pero de alguna manera deben concluir con números de estación diferentes. Esto se trata automáticamente mediante una rutina en la línea 2000. Cuando el programa se ejecuta (RUN), las dos máquinas intentarán ser la estación 1 en el canal de "transmisión" público.

La máquina que primero empiece a ejecutar el programa se convertirá en la estación 1 y a la otra le corresponderá ser la estación 2. Este sistema funciona bien en batallas navales. El programa entonces da por sentado que quien sea que esté en la estación 1 es el jugador 1 y, por consiguiente, le permite disparar primero. Sin embargo, si los dos programas se inician separados por una fracción de segundo, los dos mensajes "Soy la estación número 1" colisionan y el sistema ZX Net dejará de funcionar hasta que los jugadores pulsen la tecla BREAK.

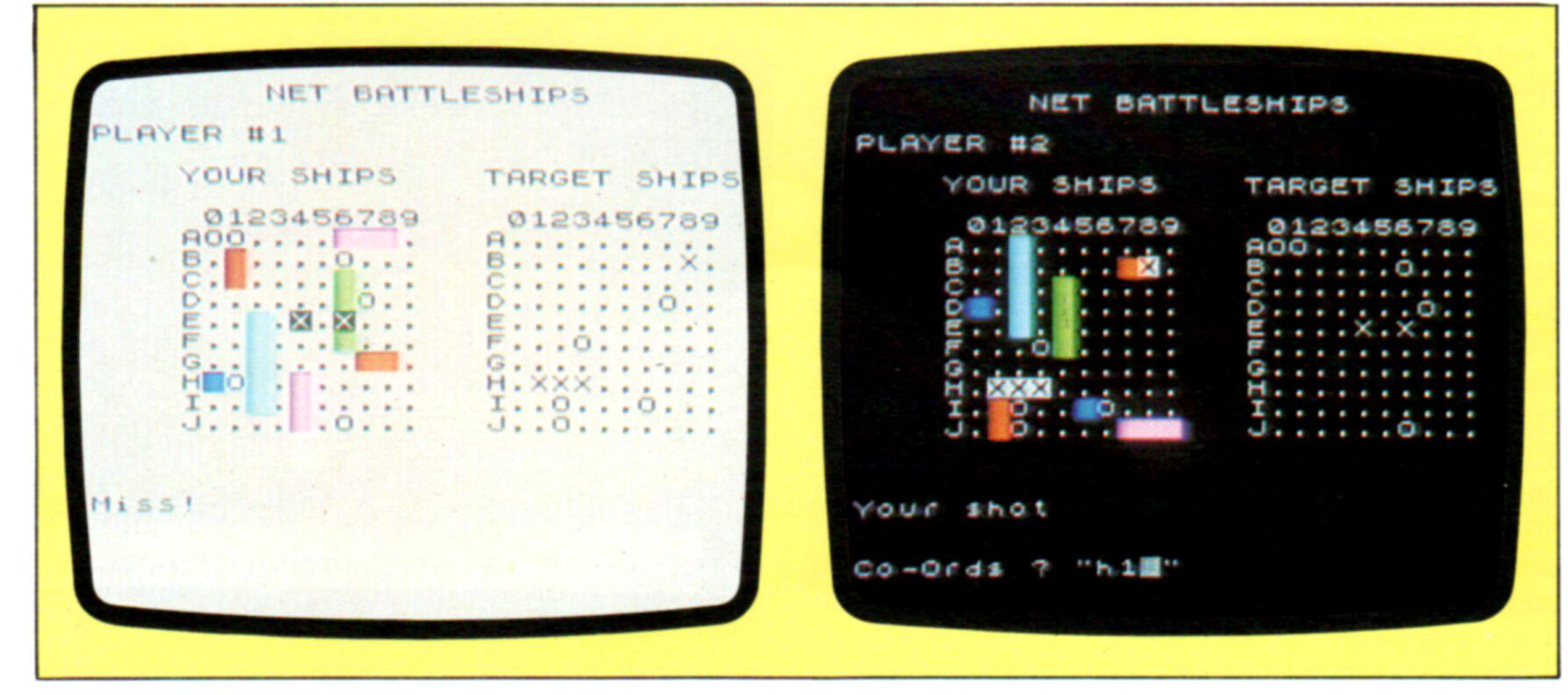
Una vez que se sabe quién es quién, para el programa es fácil comunicarse con su número oponente a través de la red. Mientras un jugador está escogiendo en su máquina un cuadrado al cual disparar, el otro está esperando conocer su elección. Los ordenadores intercambian sus funciones. Una máquina calcula los resultados del disparo y devuelve un mensaje, mientras la otra espera recibir el resultado y actualizar su visualización en pantalla en consecuencia.

Siempre y cuando se logre que los dos programas se complementen (uno enviando, el otro recibiendo), el juego se desarrollará con fluidez y sencillez. No es necesario preocuparse por la sincronización, por enviar mensajes demasiado tarde ni por perderlos después de haberlos enviado, ya que el ZX Net se interrumpe hasta que las dos estaciones están preparadas y luego transmite los datos. De modo que no tiene importancia que un jugador sea demasiado lento al seleccionar un objetivo ni que el programa tarde mucho tiempo en actualizar su pantalla.

Hay otro punto que es interesante destacar: la cantidad de datos que se transmiten se debe mantener en un mínimo. Sin embargo, no hay necesidad de enviar grandes bloques de datos. Siempre y cuando ambos programas sepan lo que significa la información, usted se puede comunicar utilizando códigos cortos. En las batallas navales, el programa devuelve el resultado de un disparo como una serie de dos caracteres. El primer carácter es un código:

1 Agua
2 Barco tocado
3 Barco tocado y hundido
4 Barco tocado y hundido y juego ganado

El segundo carácter es la clase de barco que ha sido tocado (o un cero para agua). El programa al otro lado puede decodificar esta información y visualizar mensajes adecuados. Este método hace que el tiempo que ocupa ejecutar cada jugada sea tan breve, que no parece en absoluto que haya otro ordenador implicado en el asunto.



Flotas en acción
La flota de cada jugador está formada por barcos de diferentes tamaños (desde lanchas torpederas hasta portaaviones) situados en cualquier lugar de la cuadrícula. La pantalla visualiza la posición y el estado de los barcos de un jugador y los disparos que el oponente ha efectuado contra ellos, además de la posición y el efecto de sus disparos contra la escuadra del enemigo: "X" indica tocado, "O" indica agua

Liz Heaney

Desarrollo del juego

Este programa requiere un procedimiento de comienzo ligeramente complicado, porque se ejecuta por separado en dos Spectrum. Se debe cargar una copia del programa en cada ordenador. Las dos se deben cargar desde cassette, pero es mucho más rápido cargar un Spectrum desde cassette (o microdrive) y después transmitirle el programa a la otra máquina a través de la red.
Para hacer esto, digite LOAD * "n";0 en el receptor y SAVE* "n";0 en la máquina que tiene el programa en memoria. A continuación, los jugadores han de decidir cuál de ellos va a disparar primero. Este jugador debe lanzar (RUN) el programa un poco antes que el segundo jugador. El programa luego les asigna números de red a ambas máquinas y averigua qué copia del programa está jugando en primer lugar y cuál en segundo.
Cuando empieza el programa, los dos jugadores deben fijar las posiciones de sus barcos. Esto se realiza especificando la situación de un extremo de cada navío en la cuadrícula del juego de 10 × 10 y diciendo si el resto del barco está hacia arriba, hacia abajo, a izquierda o derecha de esa posición. Esto suena complicado pero en la práctica es conveniente. Cada jugador posee dos lanchas torpederas (MTB) (1 cuadrado de longitud), dos cruceros (2 cuadrados de largo), dos acorazados (3), un destructor (4) y un portaaviones (5).
Los jugadores entonces se turnan para disparar contra un cuadrado de la cuadrícula del otro y el programa calcula el resultado de cada disparo. Gana quien antes destruye todos los barcos del oponente. Para volver a jugar, ambos jugadores tienen que digitar RUN, recordando que el que desee empezar primero debe pulsar RUN antes que el otro

```
10 REM juego de la batalla naval por red
11 REM
12 REM 2 Spectrum con Interface 1
13 REM Junio/84/Version 1.6
15 REM iniciar todo
30 GO SUB 2000: REM lucha por la red
40 LET s$ = "                                  ": REM 34 espacios
50 DIM s(8): REM tipos de barcos
60 DIM n$(8,12): REM nombres barcos
70 DIM a(10,10): REM papel cuadriculado!
80 FOR i = 1 TO 8: READ s(i),n$(i): NEXT i
90 DATA 1,"MTB",1,"MTB",2,"Crucero",2,"Crucero"
100 DATA 3,"Acorazado",3,"Acorazado",4,"Destructor",
    5,"Portaaviones"
110 LET sc = 8
120 GO SUB 3000: REM inic pantalla
130 GO SUB 4000: REM colocar barcos
140 REM ahora saltar segun que
150 REM jugador seamos. #1 dispara primero
160 IF sta = 2 THEN GO TO 400
200 REM ***Disparar!
210 LET m$ = "Su disparo": GO SUB 6000
220 GO SUB 7000: IF e = 1 THEN GO TO 210
230 OPEN #4;"n";el
240 PRINT #4;a$
250 CLOSE #4
260 REM esperar & obtener resultado
270 OPEN #4;"n";el
280 INPUT #4;a$
290 CLOSE #4
300 LET r = VAL (a$(1 TO 1)): LET x = VAL (a$(2 TO))
310 LET r = 1 THEN LET m$ = "Agua!":PRINT AT
    6 + q,18 + p;"0";:GO SUB 6000
320 IF r > 1 THEN LET m$ = "Tocado!": PRINT AT
    6 + q,18 + p;"X";: GO SUB 6000
330 IF r > 2 THEN LET m$ = "Ha hundido un "+ n$(x) + "enemigo":
    GO SUB 6000
340 IF r = 4 THEN LET m$ = "Felicitaciones ... ha ganado!!!":GO SUB
    6000: STOP
400 REM ***Disparo del enemigo
410 LET m$ = "Enemigo disparando": GO SUB 6000
420 OPEN #4;"n";el
430 INPUT #4;a$
440 CLOSE #4
450 LET p = VAL (a$(2 TO)) + 1: LET q = CODE (a$)-64
460 LET m$ = "El enemigo dispara a "+ a$: GO SUB 6000
470 LET x = a(p,q): LET a(p,q) = 0
480 IF x = 0 THEN LET r = 1: GO TO 530
490 LET r = 2
500 LET s(x) = s(x)-1
510 LET s(x) = 0 THEN LET r = 3:LET sc = sc-1
520 IF sc = 0 THEN LET r = 4
530 LET a$ = STR$ (r) + STR$ (x)
540 OPEN #4;"n";el
550 PRINT #4;a$
560 CLOSE #4
570 IF r = 1 THEN LET m$ = "Es agua": PRINT AT 6 + q,4 + p;"0";
580 IF r > 1 THEN LET m$ = n$(x) +"tocado": PRINT AT
    6 + q,4 + p;"X";
585 IF r > 2 THEN LET m$ = m$ +"y hundido"
587 GO SUB 6000
590 IF r = 4 THEN LET m$ = "Lo siento...ha perdido": GO SUB 6000:
    STOP
600 GO TO 210
2000 REM ***decidir quien es quien
2005 CLOSE #4
2010 OPEN #4;"n";0
2020 PRINT #4;"1"
2030 CLOSE #4
2040 OPEN #4;"n";0
2045 INPUT #4;a$
2050 CLOSE #4
2060 IF a$ = "1" THEN OPEN #4;"n";0: PAUSE 5:PRINT #4;"2": LET
     sta = 1
2070 IF a$ = "2" THEN LET sta = 2
2080 CLOSE #4
2090 FORMAT "n";sta: LET el = 3-sta: RETURN
3000 REM ***preparar pantalla
3010 LET col = 0: IF sta = 2 THEN LET col = 7
3020 PRINT;: BORDER 7-col: PAPER 7-col: INK col: CLS
3030 PRINT TAB4;"BATALLAS NAVALES RED"
3040 PRINT: PRINT "JUGADOR#";sta
3050 PRINT: PRINT "      BARCOS SUYOS       BARCOS ENEMIGOS"
3060 PRINT: PRINT "      0123456789       0123456789"
3070 FOR i = 1 TO 10
3080 PRINT "      ";CHR$ (i + 64;"..........       ";CHR$
     (i + 64);".........."
3090 NEXT i
3100 RETURN
4000 REM ***Preparar barcos
4010 LET m$ = "Por favor coloque sus barcos";GO SUB 6000
4020 FOR s = 1 TO sc
4030 LET m$ = STR$ s + "." + n$(s) + "long." + STR$ (s(s)):GO
     SUB 6000
4050 GO SUB 7000: IF e = 1 THEN GO TO 4030
4055 IF s(s) = 1 THEN LET xd = 0: LET yd = 0: GO TO 4130
4070 INPUT "A, B, I o D?";a$: LET xd = 3: LET yd = 3
4080 IF a$ = "A" OR a$ = "a" THEN LET xd = 0: LET yd = -1
4090 IF a$ = "B" OR a$ = "b" THEN LET xd = 0; LET yd = 1
4100 IF a$ = "I" OR a$ = "i" THEN LET xd = -1: LET yd = 0
4110 IF a$ = "D" OR a$ = "d" THEN LET xd = 1: LET yd = 0
4120 IF xd = 3 AND yd = 3 THEN GO TO 4070
4130 LET l = s(s): LET x = p: LET y = q
4140 IF x < 1 OR x > 10 OR y < 1 OR y > 10 THEN LET m$ = "Alejar
     el barco del borde": GO SUB 6000: GO TO 4030
4150 IF a(x,y)<>0 THEN LET m$ = "Por favor vuelva a colocar el
     barco": GO SUB 6000: GO TO 4030
4160 LET x = x + xd: LET y = y + yd
4170 LET l = l-1: IF l > 0 THEN GO TO 4140
4180 LET l = s(s): LET x = p: LET y = q
4190 LET a(x,y) = s: INK s(s): PRINT AT 6 + y,4 + x;" ";: INK col
4200 LET x = x + xd: LET y = y + yd
4210 LET l = l-1: IF l > 0 THEN GO TO 4190
4220 NEXT s
4230 LET m$ = "Listos para la acción!!!": GO SUB 6000
4240 RETURN
6000 REM ***IMPRIMIR m$
6010 PRINT AT 20,0;s$;AT 20,0;m$: PAUSE 100: RETURN
7000 REM ***Verificar coordenadas
7010 LET e = 0
7015 INPUT "Coord.?";a$
7020 IF LEN a$<>2 THEN LET e = 1: GO TO 7100
7030 FOR i = 1 TO 2
7040 LET c = CODE (a$(i TO i)): IF c > = 97 AND c < = 122 THEN
     LET a$(i TO i) = CHR$ (c-32)
7050 NEXT i
7060 LET q = CODE (a$(1 TO 1)): LET p = CODE (a$(2 TO 2))
7070 IF q < 65 OR q > 74 THEN LET x = q: LET q = p: LET p = x
7080 IF q < 65 OR q > 74 THEN LET e = 1
7090 IF p < 48 OR p > 57 THEN LET e = 1
7100 IF e = 1 THEN LET m$ = "Por favor vuelva a entrar las coord.":
     GO SUB 6000: RETURN
7110 LET q = q-64: LET p = p-47
7120 RETURN
```

Test en cascada (1)

En esta ocasión efectuaremos una comparación entre un valor variable y unos valores constantes

Un test en cascada consiste en una serie de preguntas continuas eliminatorias que tienen como finalidad comparar el contenido de una variable con unos valores constantes, para seguir por una u otra vía de salida de la secuencia en caso de igualdad. Tal vez el ejemplo más sencillo sea el de saber en qué día de la semana nos encontramos a base de preguntarnos el número de orden que ocupa (figura 1). Puede observarse que sólo se han empleado seis preguntas. Ello se debe a que la séptima, en este caso, es innecesaria.

Véase a continuación (figura 2) un supuesto en el que una máquina pesa unos cojinetes que llegan mezclados hasta ella, luego de haber pasado previamente por otras máquinas. Dicha máquina pesa los cojinetes uno por uno. Si su peso es de 10 g, los envía al almacén 1; si pesan 25 g, los remite al almacén 2, y si su peso es de 50 g, los envía al almacén 3. Aquellos cojinetes que no se ajusten a los pesos prefijados, son desechados y desviados a un contenedor especial, destinado a los productos defectuosos.

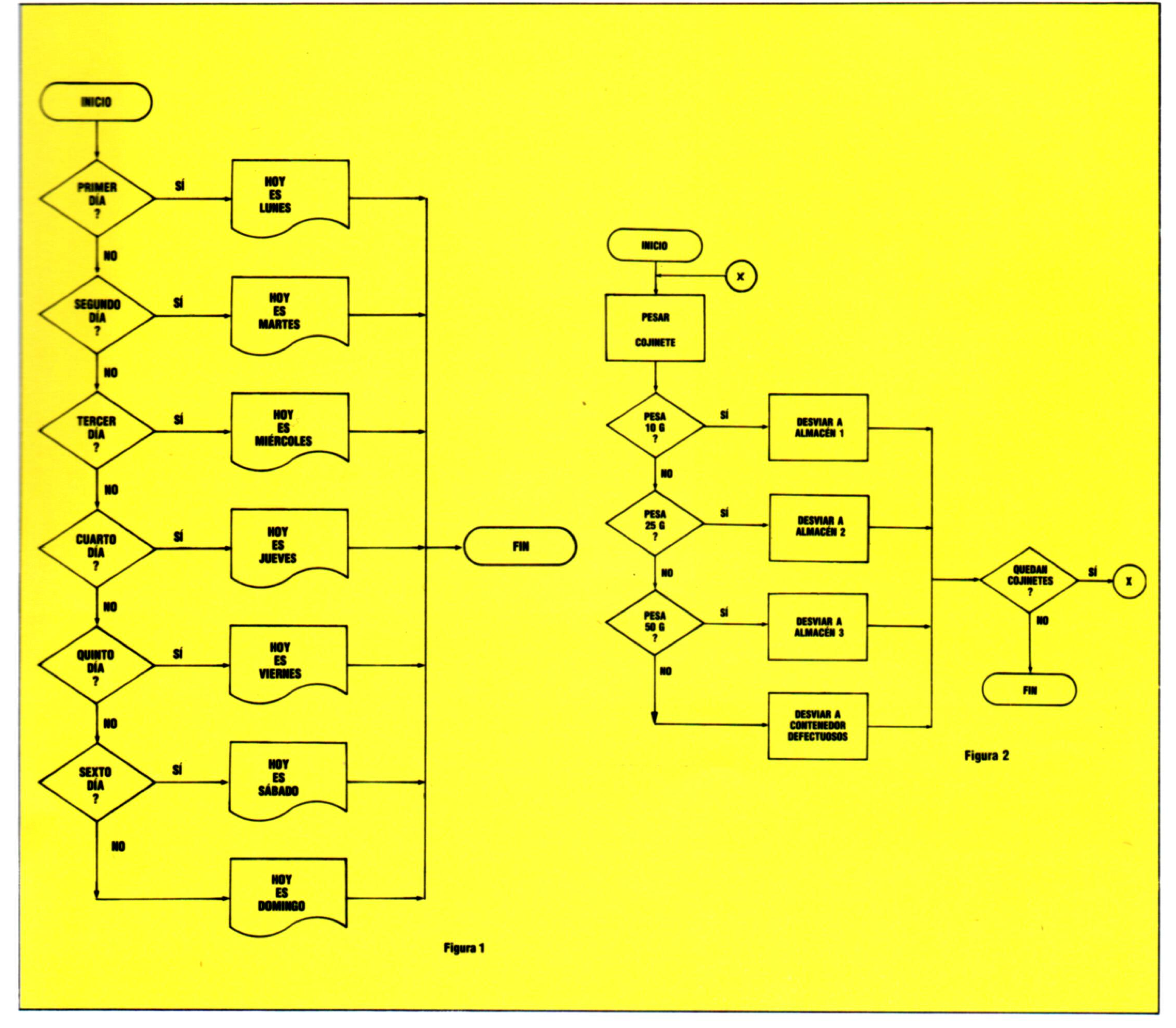


Figura 1

Figura 2



Un micro avanzado

El micro personal Advance 86 permite como pocos la posibilidad de transformarse en una máquina de oficina



Advance 86a
El Advance 86 se vende en dos versiones: el Advance 86a, que vemos en la fotografía y que se comercializa como ordenador personal, y el Advance 86b, micro de oficina compatible con el IBM

Chris Stevens

El Advance 86 se suministra en dos configuraciones: el 86a, microordenador personal basado en cassette cuyo precio está al nivel del BBC Micro pero que puede presumir de un total de 128 Kbytes de RAM, y el 86b, una máquina similar, aumentada por unidades de disco gemelas de 5 ¼" y con un BASIC más potente. Los usuarios del 86a lo pueden ampliar al nivel del 86b con sólo acoplarle un "paquete de ampliación" que contiene las unidades. El resultado es una máquina de oficina compatible con el IBM, por la mitad del precio del IBM PC.

El 86b se fabrica en dos partes: el teclado y la caja del microprocesador. Este último es considerablemente más grande de lo que sería necesario, midiendo 520 × 400 × 95 mm, y viene dentro de una carcasa de plástico negro con una puerta de metacrilato ahumado estilo equipo de alta fidelidad, que permite alojar en su interior el teclado. Todos los conectores se montan en esta unidad y el transformador de potencia está ubicado dentro de ella.

El teclado se conecta a la unidad del procesador mediante un cable coaxial y un conector DIN de cinco patillas, sobre el cual está el interruptor on/off con un indicador LED. El teclado es, por cierto, el mejor de todos los ordenadores personales existentes y es considerablemente mejor que los de la mayoría de las máquinas de oficina. Posee 84 teclas dispuestas en tres grupos: las principales teclas alfanuméricas, un conjunto de 10 teclas de función y un teclado numérico que desempeña también la función de un amplio conjunto de controles del cursor.

Las teclas de función proporcionan algunas de las funciones que se utilizan más comúnmente (RUN, LIST, SAVE, LOAD, etc.). Es muy fácil modificar estas funciones; a cada tecla se le asigna una serie de hasta 15 caracteres para identificar la instrucción, y la línea inferior de la pantalla visualiza una etiqueta de seis letras para cada una. El teclado numérico se controla mediante un interruptor de palanca rotulado como "Num Lock". En uso normal, el teclado actúa como una calculadora, pero la pulsación de Num Lock proporciona un efecto totalmente diferente: las teclas controlan entonces el cursor, permitiendo desplazarlo por la pantalla con precisión y velocidad.

En el interior de la caja del microprocesador hay una placa de circuito impreso relativamente pequeña que contiene el microprocesador de 16 bits Intel 8086 (compatible con el 8088 del IBM, pero más rápido) y 128 Kbytes de RAM. También están instalados los conectores que permiten doblar la RAM, aunque la memoria disponible para BASIC se limita a 62 Kbytes. Esto apenas si es un inconve-



Teclado inteligente
Se dice que el IBM Personal Computer cuenta con el mejor teclado entre los micros existentes. El teclado del Advance imita en gran medida el diseño de éste, aunque, de manera insólita, posee dos teclas Return y algunas teclas están cambiadas de sitio

niente, porque esta cantidad es más que suficiente para la mayoría de las aplicaciones.

El Advance está muy bien provisto de conectores e interfaces. Entre éstos se incluyen un conector especial que hace posible que un televisor o una pantalla funcionen con la fuente de alimentación eléctrica del ordenador; un conector que permite utilizar como visualización un televisor; tomas de video compuesto y RGB para monitores de video compuesto o bien de RGB (*red, green, blue:* rojo, verde, azul); una interface Centronics estándar para conexión a una impresora en paralelo; dos puertas para palanca de mando y un conector DIN de cinco patillas para utilizar con una grabadora de cintas. Cuando se usa la ampliación, se dispone de un conector RS232 para conectar una impresora en serie o un modem. Todas las interfaces aceptan cables de tipo IBM.

En modalidad de textos, el Advance visualiza 25 líneas de 40 caracteres o bien una pantalla del tipo IBM de 80 × 25; esta última apenas si es legible a menos que se emplee un monitor. La línea inferior de la pantalla normalmente visualiza rótulos de teclas de función, pero también se pueden apagar para disponer de la pantalla completa. En esta modalidad se pueden utilizar 16 colores (ya sea fijos o intermitentes). En resolución media la pantalla admite cuatro colores, con una visualización de gráficos de 320 × 200 pixels o texto en formato de 40 × 25. La modalidad en alta resolución ofrece una visualización en blanco y negro de 640 × 200 pixels o texto de 80 × 25. En la RAM se pueden almacenar y recuperar al instante siete pantallas completas de 40 × 25, lo que es muy cómodo para menús, páginas de instrucciones, etc. En la modalidad de 80 columnas se pueden almacenar y recuperar cuatro pantallas. A pesar del gran número de colores disponibles, su empleo está sometido a algunas molestas restricciones. En modalidad de textos el fondo se limita a uno de ocho colores, si bien para el primer plano y el margen se puede utilizar la gama completa. En resolución media se pueden visualizar cuatro colores, pero éstos no se pueden escoger de entre la gama completa sino que se debe seleccionar uno de dos grupos (o "paletas").

Las órdenes para gráficos del 86a se limitan a PSET (que establece el color de un pixel individual de la pantalla) y LINE (una rápida orden para trazado de líneas o recuadros). Existen instrucciones más útiles, como CIRCLE, PAINT (para rellenar con color cualquier forma en la pantalla), DRAW (que permite definir y dibujar cualquier forma) y GET y PUT (que hacen posible copiar zonas de la pantalla para gráficos en matrices, que luego se devuelven a la pantalla con tamaños o colores diferentes), pero éstas sólo se proporcionan en el Disk BASIC del 86b. Es una lástima, porque el deseo lógico de un usuario de ordenador personal sería contar con un juego completo de órdenes para gráficos. El juego de caracteres del Advance incluye la gama ASCII normal, junto con símbolos matemáticos y gráficos de bloques que permiten dibujar palos de la baraja, notas musicales, letras griegas, etc., y crear caracteres definidos por el usuario.

El resto del BASIC Advance casi no merece ningún reparo. Aunque carece de algunas de las facilidades "estructuradas" del BASIC BBC, es rápido y muy fácil de usar. Entre las características útiles se incluyen numeración y renumeración automática de líneas, PRINT USING, que permite formatear fácilmente las visualizaciones en pantalla, y la orden SWAP, que permite intercambiar los valores de dos variables. Las facilidades de sonido son buenas, aunque no asombrosas, pero nuevamente se requiere el Disk BASIC para obtener el juego completo de instrucciones. La operación de la cassette es directa: los programas en BASIC y en lenguaje máquina se cargan con una orden LOAD y los programas se ejecutan automáticamente después de cargarlos si se les agrega a la instrucción la letra "R".

Una de las características más impresionantes del Advance es el editor de pantalla. Utilizando la tecla Num Lock para que el teclado numérico actúe como control del cursor, el usuario puede desplazar éste libremente por la pantalla, haciendo correcciones e inserciones en cualquier punto.

En conjunto, el Advance parece cumplir su promesa de proporcionar un ordenador personal que se pueda mejorar para alcanzar un status completo de gestión. Indudablemente, las facilidades que ofrece el Disk BASIC del 86b, más amplio, son considerablemente superiores a las que suministra la máquina más económica, pero el 86a puede, en efecto, resistir la comparación con cualquier micro. Puede que el BASIC no esté a la altura del estándar del BBC, pero su excelente teclado y su inmensa memoria hacen que el Advance sea una opción más atrayente por el mismo precio.

Interface para impresora Centronics

Salida para RGB
También se puede utilizar una pantalla a color de gran calidad

Salida para video compuesto
Permite emplear una pantalla en color o monocromático

Modulador y salida para TV
Para poder utilizar un aparato de televisión normal

Chips ULA Ferranti
El costo del ordenador se mantiene reducido mediante la combinación de muchos circuitos en nueve chips ULA (matriz lógica de uso general) diseñados especialmente



Sistema compatible

El principal atractivo del Advance es que se lo puede adquirir como un ordenador personal y después ampliarlo hasta un nivel similar a las especificaciones de uno de oficina.
El sistema ampliado es casi por completo compatible con el IBM. En la fotografía vemos al Advance ejecutando el Flight Simulator (simulador de vuelo) de Microsoft, que se considera como la prueba definitiva para demostrar la compatibilidad. El kit de ampliación se instala en la parte superior de la unidad principal del 86a. Debe ser instalado por un distribuidor



Teclado tipo IBM

Palanca de mando y convertidor A/D
Este conector lee voltajes variables y los convierte a niveles digitales (analógico a digital). Se suele emplear para palancas de mando

Conector para teclado
El teclado, separado, se conecta aquí

Interruptor on/off

Interface para cassette
Permite utilizar una grabadora de cassette normal para guardar datos

Conectores para ampliación
El Advance 86a se amplía hasta convertirse en el 86b compatible con IBM acoplando una unidad mediante estos conectores

Microprocesador 8086
Este auténtico microprocesador de 16 bits es más potente que el del IBM PC

RAM
Además de 128 K de RAM, el Advance posee memoria extra de "control de paridad". Asimismo, hay conectores para 128 K adicionales de memoria. La memoria se puede aumentar a 256 K enchufando chips de RAM en estos conectores

Microprocesador 8087
Aquí se puede agregar el microprocesador de matemáticas de gran velocidad

Altavoz

ADVANCE 86A

DIMENSIONES
95 × 400 × 520 mm

CPU
Intel 8086, 4,77 MHz

MEMORIA
128 K de RAM (ampliables a 256 K), 64 K de ROM

PANTALLA
25 filas de 40 columnas o 25 filas de 80 columnas de texto, gráficos de 320 × 200 (4 colores) o 640 × 200 (blanco y negro). Dieciséis colores en modalidad de texto

INTERFACES
Monitores de RGB y video compuesto, palancas de mando (2), interface Centronics para impresora en paralelo, puerta para cassette, conector con toma de corriente. Puerta RS232 (sólo en el 86b)

LENGUAJES DISPONIBLES
BASIC en ROM (86a), Disk BASIC (86b)

TECLADO
Tipo máquina de escribir, 84 teclas, incluyendo 10 teclas de función y teclado numérico

DOCUMENTACION
Se suministra con guía para el usuario; también disponible un manual de BASIC. Adecuada pero no abunda en detalles

VENTAJAS
Teclado y editor de pantalla excelentes. El Disk BASIC es muy amplio y dispone de más memoria que ninguna otra máquina personal. La compatibilidad con IBM significa que puede contar con una considerable cantidad de software

DESVENTAJAS
El BASIC en ROM carece de instrucciones para hacer un mejor uso del sonido y los gráficos. Existen limitaciones para los colores en las modalidades para gráficos

Chris Stevens

De la Tierra a la Luna

"Lunar lander" es un juego sutil y refinado que contrasta con los ruidosos y frenéticos juegos actuales

En el juego *Lunar lander* hay que guiar una nave en el aterrizaje sobre la superficie de la Luna (o algún otro planeta). El ordenador de la nave no funciona, de modo que el jugador debe hacer aterrizar la nave con sumo cuidado mediante breves explosiones de su motor cohete. Obviamente, se debe tocar la superficie a una velocidad razonable y el juego implica un delicado equilibrio entre dejarse caer demasiado rápido y gastar la limitada cantidad de combustible evolucionando por la superficie del satélite.

El elemento esencial del juego es que en realidad se trata de una simulación. Si se lo programa de modo que disparar el cohete durante dos segundos siempre reste 10 km/h a su velocidad de descenso, entonces dominar el juego es demasiado sencillo. La idea es que el programa refleje el verdadero comportamiento de una nave espacial en condiciones lo más reales posible. Como resulta evidente, las operaciones matemáticas que hay que efectuar para conseguir esto son muy complejas, de manera que el programa que proporcionamos aquí es una versión simplificada.

Analicemos con mayor profundidad el problema del alunizador:

- El planeta sobre el que uno está descendiendo tiene cierta gravedad. Ésta hará que la nave espacial experimente una aceleración hacia el planeta a medida que vaya descendiendo.
- La nave espacial posee un motor cohete que contrarrestará los efectos de la gravedad empujando la nave hacia arriba.
- La nave espacial posee una masa (o peso). Cuanto mayor sea la masa, menor será el efecto del motor cohete al empujar la nave hacia arriba. La masa de la nave espacial representa su propio peso más el peso del combustible que transporta. A medida que se va consumiendo combustible, la nave espacial se va volviendo más ligera.

Por consiguiente, para reflejar la forma en que se comporta el alunizador necesitamos un conjunto de ecuaciones que implican aceleración, masa, velocidad, etc. Éstas pueden ser muy sencillas o muy



Atracción planetaria
Las cifras reseñadas representan la atracción gravitacional aproximada del Sol, la Luna y los planetas del sistema solar expresados en metros por segundo al cuadrado. Estos valores para la variable *g* del programa deben entrarse en la línea 20. Hay, por supuesto, ciertos cuerpos celestes sobre los que aterrizar sería absurdo o mortal (como el Sol, Júpiter y la Tierra), pero quizá, al efecto del juego, fuera comprensible ignorar estas cuestiones



complicadas, según lo detallistas y exactos que deseemos ser. Hemos procurado mantener estas ecuaciones en un nivel de relativa sencillez.

El principal dato que necesitamos conocer es la altura de la nave espacial. Es evidente que la nave se mueve de manera continua, ya sea cayendo debido a la gravedad o acelerando fuera del planeta debido a un uso excesivo del motor cohete. Para poder calcular dónde se halla la nave en un momento dado, se divide el "tiempo" en una serie de pasos o períodos.

En cada período se puede calcular hasta dónde se ha desplazado la nave, cuál es el cambio en cuanto a velocidad y masa, etc. Estos períodos pueden ser de la duración que se desee: cuanto más cortos sean, más exacta será la simulación. Una vez introducida la idea de los períodos, escribir las ecuaciones es fácil.

La velocidad se mide en base a unidades por hora. En dos horas, un coche que viaja a 10 km/h recorrerá 20 kilómetros. En tres horas, recorrerá 30, y así sucesivamente. Esto nos da la fórmula:

Distancia = Tiempo × Velocidad

De modo que en cada período podemos calcular lo que se ha desplazado el alunizador, hacia arriba o hacia abajo, multiplicando su velocidad por la duración del período (que definimos como unidad). Podemos entonces ajustar la velocidad acelerando la nave en función del empuje gravitacional del planeta y desacelerándola en función del empuje de los motores cohete.

La aceleración debida a la gravedad siempre es constante (la variable g del programa) y dependerá de a qué planeta se esté uno acercando. La ilustración refleja los valores para los planetas de nuestro sistema solar, pero se puede experimentar con otros valores o hacer que el programa genere g al azar para dotar al juego de mayor dificultad.

Simular el motor cohete es algo más complicado. En esta versión, el jugador puede quemar entre una y nueve unidades de combustible en un período dado, y el programa calcula la aceleración resultante, teniendo en cuenta la masa de la nave. La fórmula exacta depende de la potencia de los motores cohete y del tipo de combustible utilizado. En este programa las cifras se han elegido para que ganar el juego sea más laborioso; trate de alterarlas para ver cómo ello incide en el juego.

Un requisito que se le puede agregar al juego es que deba jugarse en *tiempo real*. Ésta es una frase de la que se abusa mucho y, en la actualidad, significa que el programa (juego, simulación, etc.) se ejecuta de forma continua, sin detenerse nunca para esperar la entrada de datos u órdenes. Con frecuencia esto no es más que la diferencia entre emplear una orden INPUT para obtener información y utilizar INKEY$ o GET.

Obviamente, *Lunar lander* es mejor, como juego, si el programa no produce ninguna interrupción para calcular cuánto combustible quemar. Si así lo hiciera, el jugador tendría tiempo para considerar la situación, efectuar algunos cálculos, etc. En la vida real, todo sería cuestión de un pensamiento más rápido.

En nuestro programa del alunizador, tenemos un bucle que se ejecuta una vez para cada intervalo de tiempo del programa. Ajustando el período de modo que sea realmente el tiempo que se tarda en ejecutar el bucle, la simulación opera en tiempo real. En una simulación hacer aterrizar la nave debería tomar tanto tiempo como el que sería necesario en la vida real. Aunque esto es de desear en una simulación, puede ser bastante difícil de conseguir en un juego como éste. La simulación suele ocupar demasiado tiempo para que resulte interesante como juego.

Son muchas las mejoras que se pueden introducir al programa de aterrizaje básico. La más obvia es agregar una visualización gráfica del descenso. En este sentido, las ideas van desde un sencillo dial redondo para el altímetro hasta una vista lateral de una pequeña nave, o incluso una panorámica a escala, hacia abajo, del punto de aterrizaje. También se podría tratar de añadir un movimiento lateral de modo que la nave no sólo tuviera que bajar hacia el punto de aterrizaje sino también situarse sobre él.

En el espacio, la nave normalmente no girará hacia los lados porque no hay nada que la empuje hacia otra dirección que no sea descendente. Pero si se está aterrizando en un planeta con atmósfera, se podría añadir el problema de un viento de superficie, por ejemplo. Algunas refinadas versiones del programa incluyen varias zonas de aterrizaje, localizadas en el fondo de túneles y cráteres, de modo que el descenso exige numerosísimas maniobras.

El *Lunar lander* podría parecer algo muy anticuado comparado con los rápidos y frenéticos juegos recreativos actuales. Pero programarlo y jugar con él es, para muchas personas, el primer encuentro con una simulación por ordenador y con todo el complicado campo de hacer que los programas reflejen situaciones del mundo real. Programar un aterrizaje lunar puede ser el primer paso hacia una nueva gama de proyectos de programación.

Alunizaje

```
10 REM Juego de aterrizaje lunar
20 LET g = −1,6
30 LET t = 1
40 LET f = 1000
50 LET v = 0
60 LET h = 2000
70 LET m = 2000 + f
80 LET g = g*t
100 REM Actualizar pantalla
110 PRINT AT 0,0
120 PRINT"          Aterrizaje lunar"
130 PRINT: PRINT "Altura.......";INT h;"  "
140 PRINT: PRINT "Velocidad....";INT v;"  "
150 PRINT: PRINT "Combustible..";f;"  "
160 PRINT
165 IF h < 0 THEN GO TO 400
170 IF f < = 0 THEN LET f = 0: PRINT "***SIN COMBUSTIBLE
    ": GO TO 190
180 PRINT "Pulsar encendido cohete 0−9"
190 LET b = 0: IF f > 0 THEN LET a$ = INKEY$: IF a$ < >"" THEN
    LET b = VAL a$
200 IF b > f THEN LET b = 0
210 LET h = h + v*t
220 LET v = v + g
230 LET v = v + (b*3000)/m
240 LET f = f−b: LET m = m−b
250 FOR i = 1 TO 50: NEXT i
300 GO TO 110
400 REM Sobre la superficie del planeta
410 IF v > −10 THEN PRINT "***Aterrizaje seguro... Bien hecho": GO
    TO 500
420 IF v > −20 THEN PRINT "***CATACRAC!... Ha destruido la nave
    pero la tripulación ha sobrevivido!": GO TO 500
430 PRINT "***BUUUM!... Nave destruida... No hay supervivientes"
440 PRINT: PRINT "Acaba de crear un nuevo cráter de";INT (−v*2.1);
    "km de ancho"
500 PRINT: PRINT "Vuelve a jugar (S/N)?";
510 LET a$ = INKEY$: IF a$ = "" THEN GO TO 510
520 IF a$ = "s" OR a$ = "S" THEN RUN
530 IF a$ <> "n" AND a$ <> "N" THEN GO TO 510
540 CLS: STOP
```

Complementos al BASIC
Éste es un listado para el Spectrum: en otras máquinas no es necesario utilizar la palabra LET. En el BBC Micro, reemplazar la línea 110 por:

```
110 PRINT TAB(0,0)
```

Reemplazar INKEY$, en las líneas 190 y 510, por INKEY$(0). Reemplazar VAL a$, en la línea 190, por VAL(a$). Reemplazar INT h e INT v, en las líneas 130 y 140, por INT(h) e INT(v).
En el Commodore 64 y en el Vic-20, reemplazar la línea 110 por

```
110 PRINT CHR$(19)
```

Reemplazar LET a$ = INKEY$, en las líneas 190 y 510, por GET a$. Reemplazar VAL a$, de la línea 190, por VAL(a$). Reemplazar INT h e INT v de las líneas 130 y 140 por INT(h) e INT(v).
En el Oric Atmos, reemplazar la línea 110 por:

```
110 PRINT @0,0
```

Reemplazar INKEY$, en la línea 190 y 510, por KEY$. Reemplazar VAL a$, de la línea 190, por VAL(a$). Reemplazar INT h e INT v, de las líneas 130 y 140, por INT(h) e INT(v)

Asesor de estilo

Un programa bien documentado es capaz de indicar lo que está haciendo y de qué manera lo está realizando

Consideremos la primera versión de nuestro programa (listado 1). Es, a todas luces, un gran misterio: es difícil adivinar lo que hace. Aparte de decir que "se entran (input) dos números, son multiplicados por otros dos números, se suman entre sí los dos resultados y se imprime la respuesta", hay muy pocos indicios de la tarea exacta que realiza el código. Ahora observemos la segunda versión del programa (listado 2). Se revela el misterio. Pero no se ha agregado ningún comentario, no hay títulos del programa ni se han insertado líneas de REM ni se ha producido documentación externa.

Vale la pena analizar de forma detallada las diferencias entre estas dos versiones. En primer lugar, los números del primer listado, carentes de todo significado, se han reemplazado por nombres (UNANY y UMES). Los números cuyos valores no cambian en el curso de la ejecución del programa se denominan *constantes*. Algunos lenguajes, como el PASCAL, poseen una notación especial para las constantes (en el listado 2 las dos constantes se definen aparte de las variables), mientras que otros lenguajes, como el BASIC, no. (Las líneas 10 y 20 del programa en BASIC emplean variables para definir las constantes.) Darles nombres a las constantes sólo vale la pena si se han de utilizar con frecuencia, de lo contrario los comentarios incluidos en el programa servirán igualmente para ese cometido.

La segunda diferencia crucial es que a todos los confusos nombres de variables se les han dado nombres más largos y significativos. Los que hemos incluido aquí (NANYS en vez de A, segsedad en lugar de e, etc.) los escogimos porque cada uno de ellos posee menos de 10 caracteres de largo, y los dos primeros caracteres son suficientes para distinguirlos entre sí. La razón para este último requisito la explicaremos enseguida.

En general, es una buena costumbre dar a las variables nombres que guarden alguna relación con el papel que desempeñan en el programa. Por ejemplo, al contador de un bucle se lo podría llamar BUCLE (en vez de los nombres habituales, J o I), y los primeros y los últimos valores del contador se pondrían en constantes o variables con nombres apropiados. Por consiguiente, un bucle como éste:

```
FOR J = 1 TO 10...NEXT J
```

podría escribirse así:

```
FOR BUCLE = PRIMERO A DECIMO...NEXT BUCLE
```

Los nombres de variables largos, por supuesto, llevan más tiempo en digitarlos y ocupan más memoria, pero poseen la ventaja de hacer que los programas resulten más fáciles de comprender y aceleran el proceso de depuración (*debugging*). Si su lenguaje utiliza sólo los dos primeros caracteres de los nombres para distinguirlos entre sí, asegúrese de que los nombres que elija difieran en sus dos primeros caracteres. De no ser así, dos nombres de variables largos (p. ej., CODIGO y COMP) podrían parecerle bien distintos al programador, pero para el ordenador serían indistinguibles.

Otra importante diferencia entre los listados es que el segundo utiliza indicaciones extensas y plenas de significado para sus input y les agrega a sus salidas una explicación razonable (las líneas PRINT en BASIC, las líneas write en PASCAL). Con ello se consiguen dos cosas importantes. La primera es que el programa sea más legible. Aun cuando las variables constaran de una sola letra, el programa seguiría teniendo mucho más sentido que anteriormente. La segunda ventaja, aún más importante, es que hace que el programa sea asequible incluso para quien nunca antes lo hubiera visto.

Trazado de programa

Los usuarios del PASCAL ya serán conscientes de las ventajas de trazar un programa adecuadamente en la pantalla. Cosas muy sencillas (como indentar las líneas, dejar líneas en blanco y combinar mayúsculas y minúsculas) pueden convertir una masa impenetrable de símbolos en un trozo de lógica sensato y legible. Formatear un programa para la pantalla o la impresora realmente adquiere todo su valor cuando sus programas utilizan construcciones de bucle (FOR...NEXT, WHILE...WEND, REPEAT...UNTIL) y, en especial, cuando hay bucles anidados dentro de otros.

Habiendo dicho esto, es lamentable que la mayoría de los BASIC den muy pocas opciones sobre la forma en que uno puede trazar el programa. En este sentido, los lenguajes compilados, como el PASCAL, son mucho más flexibles, puesto que suelen escribirse con un editor de textos (o procesador de textos). Por el contrario, editar un programa en BASIC generalmente es un asunto bastante imperfecto (a menos que, como en el MBASIC de Microsoft, su intérprete tome una versión ASCII del programa y la convierta en un programa ejecutable). Lo que es aún peor, muchos BASIC toman los programas que usted escribe ¡y los vuelven a reformatear para eliminar la indentación! Hay otros que, por el contrario, agregan una indentación que uno no había incluido. El BBC Micro es bastante eficiente en este sentido, pero hay que recordar darle la orden LISTO. La mayoría de los sistemas PASCAL incluyen un formateador y por lo general son muy útiles. No obstante, en aras de su propia claridad de pensamiento, es una buena idea concebir algunas convenciones de formateado, siempre dentro de los límites de su lenguaje.

Los comentarios, por supuesto, son la forma principal de documentar los programas dentro de los propios programas. Nuevamente, las convenciones varían de un lenguaje a otro. El BASIC utiliza



la sentencia REM. La palabra REM debe aparecer al principio de cada comentario y posteriormente el intérprete ignorará todo lo que encuentre hasta el siguiente indicador de final de sentencia (: o (cr)). En otros lenguajes (PASCAL, PL/1, PROLOG, etc.) los comentarios se encierran entre /* y */ (algunas veces { y }), y el compilador hace caso omiso de todo cuanto haya entre estos signos. Una ventaja que ofrece este sistema es que los comentarios pueden ocupar más de una línea. La desventaja es que si uno se olvida del segundo */, ¡el resto de su programa se toma como un comentario y se ignora!

Utilice un comentario cada vez que piense que puede ser necesaria alguna explicación: cuando está definiendo constantes, inicializando variables, empezando un programa o un nuevo procedimiento (subrutina), definiendo una función o escribiendo algún código que, debido a su complejidad, no se entienda a simple vista. Los comentarios no han de ser extensos ni redundantes, y a menudo sólo se requiere un recordatorio. Cuando usted está intentando comprender la lógica de un programa de aventuras del año anterior, los largos bloques de comentarios generales que interrumpen el código y no proporcionan detalles suficientes pueden ser más un obstáculo que una ayuda, de modo que trate que sus comentarios sean breves y concisos. Colóquelos antes de secciones tramposas del código, e inclúyalos en éste sólo cuando no existan posibilidades de que interfieran la lectura de la estructura lógica del programa. Nuestro programa final (listado 3) ofrece algunos ejemplos.

La documentación externa, en forma de guías y especificaciones escritas, es la más dura y tediosa de realizar. Para los programadores, los estudios han demostrado que la documentación escrita se suele consultar sólo como último recurso. Sin embargo, cuando se la utiliza puede significar un considerable ahorro de esfuerzo. Si su programa no es demasiado extenso y está bien documentado internamente, es poco probable que alguna vez se encuentre con la necesidad de una documentación externa del *programa*. La documentación para el *usuario* es otra cuestión y la analizaremos en un capítulo posterior. No obstante, suele ser útil tener a mano alguna documentación escrita cuando se trata de revisar un programa antiguo o de depurar uno nuevo. Una de las formas en que los lenguajes de la llamada "quinta generación" intentan aumentar la productividad del programador es mediante la generación automática de documentación. Esto se conseguirá utilizando información desde la fase de diseño del desarrollo de un programa. No es sorprendente que una de las mejores formas de documentar sus propios programas sea aplicando esta misma técnica.

Vaya formando un archivo para sus programas a medida que los escriba. Coloque en el mismo todas las notas que tome mientras va diseñando el programa, incluyendo borradores de algoritmos y diagramas de flujo. Y, lo que es más importante, conserve la versión final del diagrama que ha utilizado para escribir la versión final. Si posee una impresora, conserve un listado del programa acabado. Observe que en nuestra versión completa del programa, el primer comentario incluye el nombre de éste y una fecha. Cada vez que modifique un programa, cambie también su fecha; así sabrá que se trata de la última versión.

Correctamente documentado

Listado 1

BASIC

(a)
```
10 INPUT A,B
20 C = A*31536000
30 D = B*2592000
40 E = C + D
50 PRINT E
```

PASCAL

(b)
```
program abcde (input,output);
var a,b,c,d,e:integer;
begin
read(a,b);
c: = a*31536000;
d: = b*2592000;
e: = c + d;
writeln(e);
end.
```

Listado 2

BASIC

(a)
```
10 UNANY = 31536000
20 UMES = 2592000
30 PRINT"Entre su edad (en formato AA,MM)";
40 INPUT NANYS,NMESES
50 ASEGS = NANYS*UNANY
60 MSEGS = NMESES*UMES
70 SEGSEDAD = ASEGS + MSEGS
80 PRINT"Su edad en segundos es (aproximadamente)":SEGSEDAD
```

PASCAL

(b)
```
program edadensegs (input,output);
const
   unany = 31536000;
   umes = 2592000;
var
   nanys,nmeses,asegs,msegs,segsedad:integer;
begin
   write("Entre su edad (en formato AA,MM");
   read(nanys,nmeses);
   asegs: = nanys*unany;
   msegs: = nmeses*umes;
   segsedad: = asegs + msegs;
   writeln('Su edad en segundos es (aproximadamente)',segsedad);
   end.
```

Listado 3

BASIC

(a)
```
10 REM "EDADENSEGUNDOS" Junio 1984
20 REM Entra edad en años y meses (AA,MM) y
30 REM utiliza una conversion aproximada (mes = 30 dias)
40 REM para dar la edad en segundos.
50 REM
60 UNANY = 31536000:REM      segundos en 365 dias
70 UMES = 2592000:REM      segundos en 30 dias
80 PRINT"Entre su edad (en formato AA,MM)";
90 INPUT NANYS,NMESES
100 REM la edad en segundos es (edad en años*segundos en el año) mas (meses desde ultimo
     cumpleaños*segundos en el mes)
110 ASEGS = NANYS*UNANY
120 MSEGS = NMESES*UMES
130 SEGSEDAD = ASEGS + MSEGS
140 PRINT"Su edad en segundos es (aproximadamente)":SEGSEDAD.
```

PASCAL

(b)
```
program edadensegundos (input,output);
/* Junio 1984
   lee edad en años y meses (AA,MM) y usa una conversion aproximada (mes = 30 dias) para dar la
   edad en segundos.                                                                          */
const
   unany = 31536000;      /* segundos en 365 dias */
   umes = 2592000;      /* segundos en 30 dias */
var
   nanys,nmeses,asegs,msegs,segsedad:integer;
begin
   write('Entre su edad (en formato AA,MM)');
   read(nanys,nmeses);
   /*la edad en segundos es (edad en años*segundos en el año)
      más (meses desde el ultimo cumpleaños*segundos en el mes)*/
   asegs: = nanys*unany;
   msegs: = nmeses*umes;
   segsedad: = asegs + msegs;
   writeln('Su edad en segundos es (aproximadamente)', segsedad;
end.
```

Guerra en familia

"Apocalypse" (Apocalipsis), el juego de la devastación nuclear, es un paquete tradicional del tipo "Monopoly", que exige aptitudes tácticas y reacciones rápidas por parte del usuario

La mayoría de los juegos por ordenador son diversiones solitarias, antisociales. Los anuncios publicitarios muestran familias enteras reunidas alrededor del ordenador personal, pero los juegos por lo general están diseñados para un único jugador.

Sin embargo, *Apocalypse* recupera la tradicional rivalidad y la intriga propias del *Monopoly* y otros juegos de mesa de este tipo y, por tanto, es especialmente apropiado para reuniones de familia. De hecho, la versión para el BBC permite la participación de hasta 15 jugadores. Se trata de un auténtico juego para varias personas, ya que no se limita simplemente a que éstas se vayan turnando para participar en un juego pensado para una sola persona.

Originalmente *Apocalypse* era un juego de tablero, basado en otro denominado *Diplomacy*, que ahora ha sido ampliado para sacar partido de las capacidades para gráficos y proceso de números del microordenador. El juego principal comprende cuatro "escenarios de guerra" (Europa, el Caribe, Gran Bretaña y Londres) y se pueden adquirir kits de ampliación para incluir acciones en Sudáfrica, Levante, el Ártico, Estados Unidos, el Sudeste asiático, la campaña del Pacífico durante la segunda guerra mundial, el espacio exterior y campañas históricas basadas en la caída del Imperio romano y las batallas napoleónicas. Estas opciones están disponibles en tres cintas que se combinan con el juego principal.

La pantalla está dividida en una serie de cuadrados, visualizados como una matriz de 40 × 20 (BBC) o 20 × 20 (Spectrum). Se utilizan cuadrados azules para representar el mar y mediante otros colores se describen distintas características geográficas: zonas rùrales o urbanas, ciudades, montañas o desiertos. También hay símbolos para representar las fuerzas oponentes. Estos gráficos dependen de las posibilidades de la máquina. Los gráficos del Spectrum, más simples, ofrecen la ventaja de la claridad: los cuadrados se rellenan o se dejan vacíos, por lo cual es fácil detectar el territorio que está ocupado por una fuerza atacante. La visualización del BBC, si bien permite apreciar más detalles gráficos, aparece como desordenada y los símbolos de los jugadores tienden a perderse en el fondo.

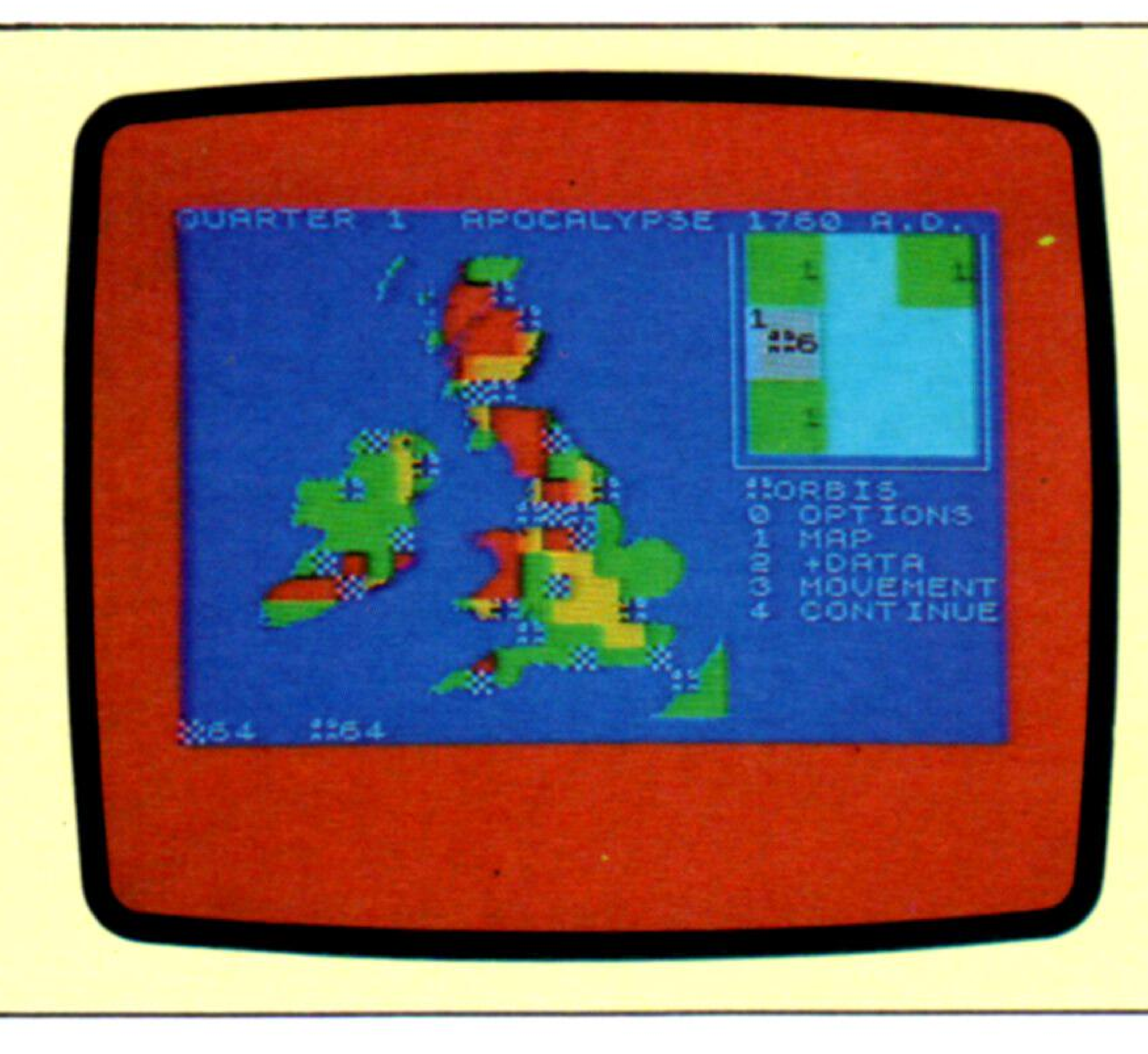
Los jugadores se turnan para desplegar sus fuerzas de la forma apropiada al escenario elegido. Las fuerzas del ejército y de la marina se desplazan por la zona de juego y lanzan o repelen ataques, mientras el ordenador actúa como árbitro. Los movimientos se realizan seleccionando opciones de diversos menús y, en este sentido, la visualización se podría mejorar para hacer que las cosas resultaran más sencillas. La opción *nuke* (ataque nuclear) se puede seleccionar en el juego principal, con consecuencias previsiblemente devastadoras, pero esta opción sólo se puede utilizar en su correcto contexto histórico: el programa no permitirá que usted lance un ataque nuclear preventivo contra legiones romanas enemigas, por ejemplo.

Éste es un juego largo pero, si se encuentran jugadores bien dispuestos, es muy interesante. Si es o no bueno para jugarlo con un microordenador, ya es otro tema. Tal vez a muchas personas el juego sobre tablero les resulte igualmente atractivo, pero por ordenador es posible que llegue a apasionar.

Apocalypse: Para el BBC Micro (2-15 jugadores) y el ZX Spectrum (2-4 jugadores)
Editado por: Red Shift, 12 Manor Road, London N16
Autores: H. Watson (BBC); R. Tyler (Spectrum)
Diseño original del juego: Mike Hayes
Palancas de mando: No se necesitan
Formato: Cassette

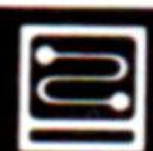
Zonas de guerra

Apocalypse es un juego para toda la familia porque pueden participar hasta 15 personas simultáneamente (en la versión para el BBC), o hasta cuatro, en la que vemos aquí, para el Spectrum. Siempre son necesarios al menos dos jugadores. Éstos se ven envueltos en una confrontación armada que a la larga puede desembocar en un conflicto nuclear. Afortunadamente, la visualización se limita a mapas como éstos y no ofrece imagen alguna de la devastación





Liz Heaney



Paradero conocido

Estudiamos una rutina en lenguaje máquina, casi imprescindible para los creadores de programas de juegos

Un juego recreativo de calidad necesita estar escrito, por lo menos parcialmente, en lenguaje máquina. Esto constituye un auténtico desafío para el principiante; por este motivo hemos decidido presentar una rutina para sprites concebida especialmente para el Spectrum.

El BASIC del Spectrum es muy limitado, y eso se nota especialmente a la hora de dar movimiento a los gráficos. Los juegos animados requieren el uso de sprites de varias formas y tamaños que han de desplazarse suavemente por la pantalla en todas las direcciones.

No es fácil escribir rutinas de gráficos en assembly, pero el programa que aquí presentamos probablemente sugerirá al lector un buen puñado de ideas con las que podrá afrontar la tarea. Este programa imprime un fondo de asteriscos colocados al azar y permite además dar movimiento a cualquier figura escogida por el usuario (nosotros elegimos una cruz) que recorrerá la pantalla con sólo tocar las teclas del cursor. La cruz se mueve paso a paso, un pixel cada vez, arriba, abajo, a derecha e izquierda, sin modificar el fondo. La rutina que mueve el sprite es fácilmente incorporable a cualquiera de sus programas en BASIC.

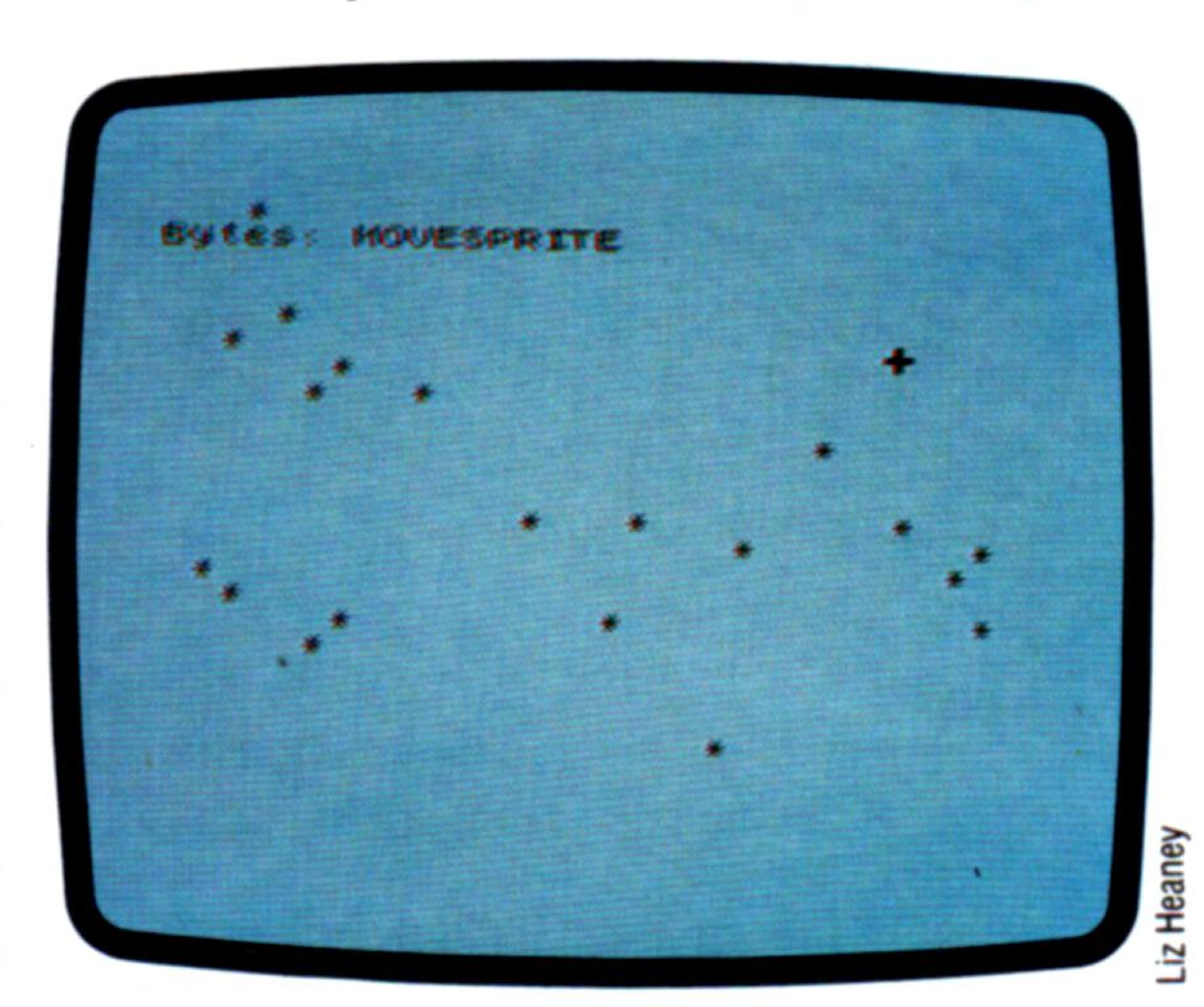
Para introducirla en un Spectrum, hay que entrar primero el programa en BASIC. Sólo entonces se puede entrar el código máquina ya sea por medio del programa de carga en BASIC, ya sea empleando un ensamblador adecuado. Es posible conservar en cinta de cassette ambos programas, el de BASIC y el de lenguaje máquina, con las líneas 9000 y 9010 del programa BASIC.

Comencemos examinando el fragmento en BASIC, con el fin de comprender plenamente el funcionamiento de todo el programa. La subrutina de la línea 1000 recoge (READ) la forma del sprite definida mediante las sentencias DATA y la coloca (POKE) en la memoria RAM donde le sea posible utilizarla al código máquina. El fondo se imprime con las líneas 90 hasta la 110, y la posición inicial de la cruz se establece en la 120. Con PRINT AT 10,16 el intérprete de BASIC calcula la dirección de la pantalla correspondiente a estas coordenadas de carácter, almacenando dicha dirección en la variable de sistema DFCC (direcciones 23684 y 23685), donde también es accesible por parte del código máquina. La línea 130 llama a la sección de inicialización del programa en código máquina. El bucle de las líneas 140 a 180 espera la pulsación de cualquier tecla, coloca (POKE) el valor numérico de la tecla pulsada en una posición de la memoria que pueda ser leída por el código máquina y posteriormente hace que este código máquina mueva el sprite un pixel en la dirección que determina la misma tecla.

El programa en assembly comienza con la definición de las posiciones de memoria empleadas. Así, KEY (*tecla*, en inglés) indica la posición donde se almacenó el valor de la tecla. SPRPOS significa la dirección de la memoria que contiene la *pos*ición del *spr*ite en la pantalla. SPRTAB es la tabla en la que el programa almacena la definición del sprite y el contenido de las posiciones de la pantalla sobre las que se superpone el sprite. Éste se mueve a cualquier punto de la pantalla; no se limita, pues, a saltar cuadrados de carácter enteros. Y sucede así porque los ocho bits de cada fila del sprite pueden dividirse en dos bytes de memoria de pantalla y la tabla emplea dos bytes para almacenar esos ocho bits, partidos igual que en la pantalla. Usamos BITPOS para almacenar el número de bits que, desde el inicio del byte, se han ido desplazando los datos del sprite.

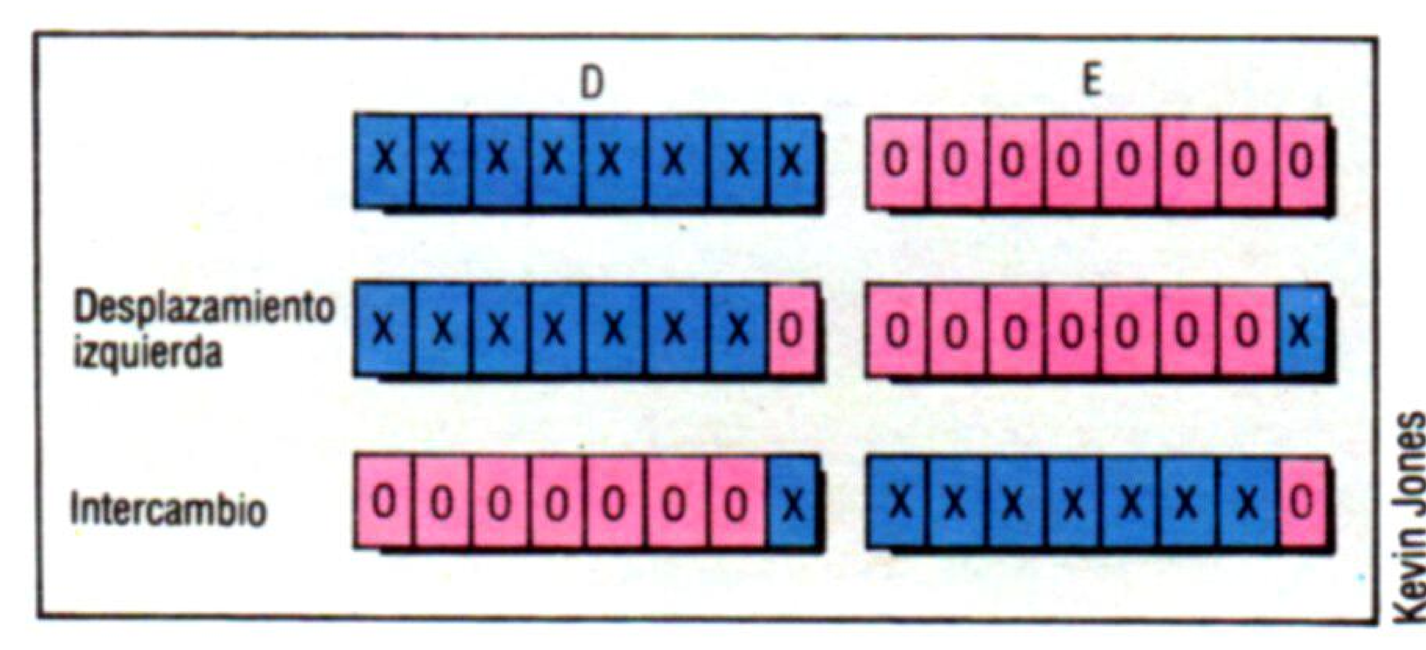
El trozo del programa que sirve para inicializar lee de DFCC la dirección inicial de la pantalla, salta a la sección etiquetada con SAVSCR (recuerde que en inglés pantalla se dice *screen*), donde guarda la dirección de la pantalla en SPRPOS y carga el valor 1 en el registro D para llamar finalmente la subrutina UNDER. Si D está en 1, la rutina UNDER copia el contenido del área de la pantalla en la cual aparecerá el sprite a fin de restaurarlo, una vez desplazado el sprite. La otra subrutina que llama ahora el programa es la PRSPRT que visualizará (PRint) el sprite en la pantalla.

Del movimiento del sprite se encarga la sección de programa que empieza en MOVSPR. Lo primero



Liz Heaney

El sprite en movimiento
El programa de demostración en BASIC desplaza el sprite (especificado como una cruz en las sentencias DATA) a través de la pantalla con un fondo de estrellas. Sólo hay que apretar las teclas del cursor



Kevin Jones

Desplazamiento e intercambio
El desplazamiento de los bytes del sprite (que representamos con las X), dentro del registro DE, queriendo obtener un efecto de movimiento a la izquierda o a la derecha, quizá cause que algún bit caiga por un extremo. En ese caso D y E se intercambian, volviendo a reunir los bits del sprite

Direccionamiento de pantalla
A efectos de representación en memoria, la pantalla de 24 líneas del Spectrum está dividida en tres secciones de ocho líneas cada una; en el dibujo inferior se muestra el detalle del direccionamiento de la sección central. Cada línea de la pantalla está dividida en ocho líneas de alta resolución de 32 bytes, dentro de los cuales cada bit representa un punto o pixel. Los bytes son numerados consecutivamente a lo largo de la fila de alta resolución; al terminar una de ellas se empieza con la misma fila de alta resolución de la línea inmediatamente inferior: por tanto, los bytes de las ocho filas superiores de las ocho líneas de cada una de las tres secciones de la pantalla tienen direcciones consecutivas. Al terminar las ocho primeras filas, se sigue numerando por el primer byte de la segunda fila de la primera línea; todos los bytes de las ocho segundas filas de alta resolución se numeran correlativamente a partir de ese byte, y así sucesivamente. La dirección del primer byte de la fila superior de la primera línea de una sección se obtiene sumando una unidad a la dirección del último byte de la octava fila de la octava línea de la sección anterior. El siguiente programa coloca un valor $FF en todos los bytes de la pantalla:

```
50 LET INIPANT = 16384
60 LET FINPANT = 22527
100 FOR B = INIPANT TO
     FINPANT
200 POKE B,255
300 NEXT B
```


Kevin Jones

que hace es poner 0 en el registro D y llamar la subrutina UNDER. Si D está en 0, UNDER devuelve a la pantalla el contenido del fondo que salvó previamente, al tiempo que borra el sprite de la pantalla. Entonces es cuando el programa toma la posición del sprite y el valor de la tecla, lo comprueba y llama a la rutina que preparará el movimiento en la dirección adecuada, saltando finalmente a SAVSCR que, como antes, nos guardará el contenido del fondo de la pantalla y visualizará el sprite.

Dos son las rutinas que se encargan del movimiento vertical del sprite, ABOVE (*arriba*, en inglés) y BELOW (*abajo*). Para entenderlas, es inevitable examinar la extraña manera como el Spectrum relaciona direcciones de memoria con posiciones de pantalla. El capítulo 24 del Manual del Spectrum la explica. Mirando las direcciones en hexadecimal, se observa que cada byte de los ocho que componen un carácter (y son 256 caracteres los que integran una sección de pantalla) tiene una dirección, compuesta a su vez de dos bytes: el byte inferior o byte *lo* coincide con el número del carácter dentro del bloque, mientras que el byte superior o byte *hi* se incrementa en una unidad cuando nos desplazamos una línea de pixels hacia abajo en la pantalla. Ésta es la razón por la que las ocho filas de pixels de un carácter poseen direcciones que van del $4000 al $47FF para el tercio superior de la pantalla, del $4800 al $4FFF para el tercio central, y del $5000 al $57FF para el inferior. (Recuerde que $ significa número en hexadecimal, aunque algunos ensambladores emplean el símbolo #).

La subrutina BELOW espera una dirección de pantalla en el registro doble HL, calcula la dirección del byte inmediatamente inferior a esta posición de la pantalla y deja el nuevo valor en HL. Observadas en binario, si las direcciones de la pantalla tienen los tres bits inferiores de H con el valor 111, la siguiente fila de pixels hacia abajo se halla en un bloque de caracteres distinto. Esto es lo que primero comprueba BELOW, y si todavía nos encontramos en el mismo bloque de carácter, sólo nos falta añadir 1 a H. Si nos hallamos en un bloque de carácter distinto, añadiremos $20 a L (puesto que hay 32 caracteres en una línea). Si el nuevo valor de L está entre 0 y $1F (tres bits altos a 0), esto significa que estamos en un bloque de pantalla distinto. En cuanto al valor de HL, éste corresponde a la dirección de pantalla en curso.

Si estamos en el mismo bloque de pantalla, tenemos además que restar 7 a H. Todo esto lo entenderá mejor si examina el código máquina y observa lo que ocurre con las direcciones mostradas en la tabla.

Por lo demás, ABOVE funciona como BELOW, sólo que su misión es calcular la dirección del pixel que está encima de la posición de la pantalla.

Las subrutinas LMOVE y RMOVE desplazan los bits del pixel como un todo a la izquierda y a la derecha. Como son también semejantes, veamos tan sólo cómo funciona LMOVE. El apuntador de la posición de los bits pasa al acumulador, siendo aquél un número de un solo byte entre el 0 y el 7 que numera a cada bit dentro del byte. Para efectuar el movimiento se resta 1 al valor actual del acumulador y el resultado también será un número entre el 0 y el 7 (a menos que el valor original del apuntador fuera 0, en cuyo caso obtendríamos el resultado de 255). La instrucción AND 7 se encarga de que el valor del acumulador quede siempre entre 0 y 7. Seguidamente encontramos un bucle para las ocho filas de pixels de un sprite. La tabla va proporcionando dos bytes cada vez, correspondientes a cada fila, que se cargan en el registro doble DE, para realizar después una rotación de 16 bits a la izquierda en el mismo DE. Si ningún bit queda fuera del extremo superior de D para colocarse en el inferior de E, entonces pondremos de nuevo en la tabla todos los bits del sprite ya desplazados y pasaremos a la siguiente fila de pixels. Pero si se trasladó un bit del sprite desde el extremo superior de D al inferior de E debemos intercambiar D y E antes de almacenarlos de nuevo en la tabla. La rutina, por último, restará 1 a HL para que el sprite aparezca en pantalla una posición a la izquierda.

La última subrutina es PRSPRT, que tiene como misión la de visualizar el sprite sobre la pantalla. La forman dos bucles anidados, uno para las ocho filas de pixels del sprite controlado por el registro C, y otro, controlado por el registro B, para los dos bytes en que se reparte la fila de pixels. El núcleo de la rutina está en su sección central, encargada de almacenar los bits del sprite en pantalla sin afectar los otros bits en pantalla previos que el sprite debe dejar intactos. La dirección de la pantalla se coloca en el registro doble HL y la dirección de la tabla del sprite en el registro IX. Nuestra PRSPRT toma un byte de la trama de pixels del sprite y le aplica la operación OR con el que ya está en pantalla, de modo que nos resulten los puntos apagados del sprite superpuestos al contenido previo de la pantalla.

Este programa no es de uso universal. Por ejemplo, el tamaño máximo de un sprite está limitado a 8 × 8 pixels, además sólo admite un sprite y éste no traslada consigo sus propios colores. Pero una vez entendido, de seguro que usted mismo sabrá añadir al programa algunos detalles. Con o sin modificaciones, se trata de un muy útil aditamento para muchos programas en BASIC y en lenguaje máquina.



Sprites con vida

```
   5 REM "Programa 1"
  10 CLEAR 45055: REM AFFF EN HEXA
  20 LOAD "MOVESPRITE"CODE
  30 LET KEY = 45056
  40 LET BITPOS = 45059
  50 LET SPRTAB = 45060
  60 LET INIT = 45312: REM B100 HEX
  70 LET MOVSPR = 45317: REM B105 HEX
  80 GO SUB 1000: REM SET UP SPRITE
  90 FOR I = 1 TO 20
 100 PRINT AT 21*RND,31*RND;"*";
 110 NEXT I
 120 PRINT AT 10,16;
 130 RANDOMIZE USR INIT
 140 LET X$ = INKEY$: IF X$ = "" THEN GO TO 140
 150 LET X = VAL X$
 160 POKE KEY,X
 170 RANDOMIZE USR MOVSPR
 180 GO TO 140
1000 POKE BITPOS,0
1010 FOR I = 0 TO 7
1020 READ X
1030 POKE SPRTAB+2*I,X
1040 POKE SPRTAB+1+2*I,0
1050 NEXT I
1060 RETURN
2000 DATA BIN 00011000
2010 DATA BIN 00011000
2020 DATA BIN 00011000
2030 DATA BIN 11111111
2040 DATA BIN 11111111
2050 DATA BIN 00011000
2060 DATA BIN 00011000
2070 DATA BIN 00011000
```

```
   5 REM "Programa 2"
  10 LET A = 45312
  20 FOR L = 1000 TO 1420 STEP 10
  30 LET S = 0
  40 FOR A = A TO A+7
  50 READ B
  60 POKE A,B
  70 LET S = S+B
  80 NEXT A
  90 READ C
 100 IF C<>S THEN PRINT "ERROR DE LINEA ";L: STOP
 110 NEXT L
 120 READ B
 130 POKE A,B
 140 READ B
 150 POKE (A+1),B
 200 PRINT "INSERTE CINTA PROGRAMA"
 250 SAVE "MOVESPRITE"CODE 45312,400
1000 DATA 42,132,92,24,44,22,0,205,561
1010 DATA 61,177,42,1,176,58,0,176,691
1020 DATA 254,5,32,5,205,165,177,24,867
1030 DATA 24,254,6,32,5,205,109,177,812
1040 DATA 24,15,254,7,32,5,205,136,678
1050 DATA 177,24,6,254,8,192,205,228,1094
1060 DATA 177,34,1,176,22,1,205,61,677
1070 DATA 177,205,35,178,201,42,1,176,1015
1080 DATA 221,33,4,176,14,8,229,6,691
1090 DATA 2,203,66,32,6,221,126,16,672
1100 DATA 119,24,4,126,221,119,16,35,664
1110 DATA 205,83,178,221,35,16,234,225,1197
1120 DATA 13,200,221,35,197,213,205,109,1193
1130 DATA 177,209,193,24,217,62,7,164,1053
1140 DATA 254,7,40,2,36,201,17,32,589
1150 DATA 0,25,62,224,165,32,4,205,717
1160 DATA 83,178,201,124,214,7,103,201,1111
1170 DATA 62,7,164,40,2,37,201,17,530
1180 DATA 32,0,167,237,82,62,224,165,969
1190 DATA 254,224,32,4,205,76,178,201,1174
1200 DATA 124,198,7,103,201,221,33,3,890
1210 DATA 176,221,126,0,61,230,7,221,1042
1220 DATA 119,0,79,221,35,6,8,221,689
1230 DATA 94,0,221,86,1,203,3,245,853
1240 DATA 203,11,241,203,18,203,19,62,960
1250 DATA 7,185,32,3,122,83,95,221,748
1260 DATA 115,0,221,114,1,221,35,221,928
1270 DATA 35,16,220,62,7,185,192,43,760
1280 DATA 205,76,178,201,221,33,3,176,1093
1290 DATA 221,126,0,60,230,7,221,119,984
1300 DATA 0,79,221,35,6,8,221,94,664
1310 DATA 0,221,86,1,203,11,245,203,970
1320 DATA 3,241,203,26,203,27,62,0,765
1330 DATA 185,32,3,122,83,95,221,115,856
1340 DATA 0,221,114,1,221,35,221,35,848
1350 DATA 16,220,62,0,185,192,35,205,915
1360 DATA 83,178,201,42,1,176,221,33,935
1370 DATA 4,176,14,8,229,6,2,221,660
1380 DATA 126,0,47,87,126,162,221,182,951
1390 DATA 0,119,35,205,76,178,221,35,869
1400 DATA 16,237,225,13,200,197,205,109,1202
1410 DATA 177,193,24,224,62,63,188,192,1123
1420 DATA 38,87,201,62,88,188,192,38,894
1430 DATA 64,201,265
```

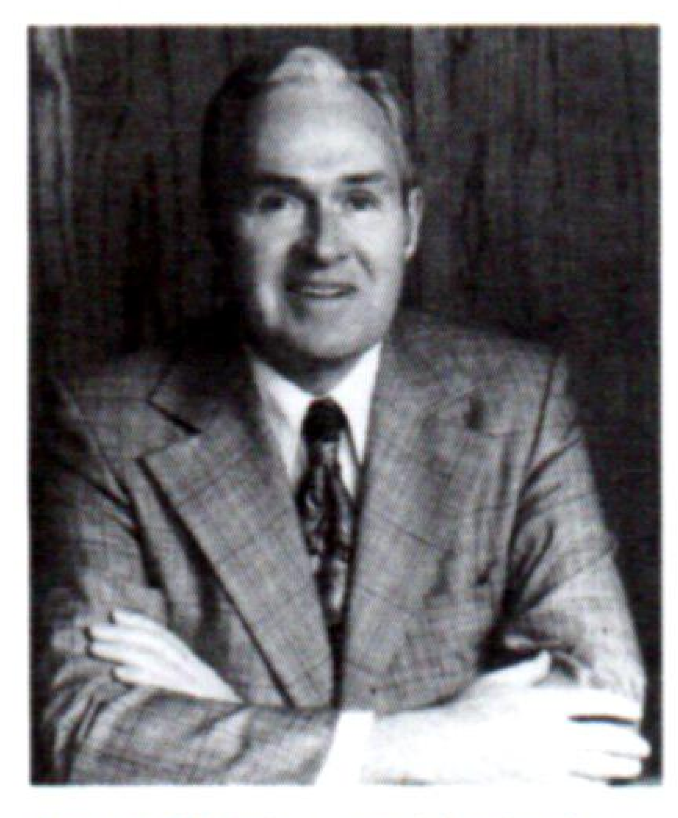
```
KEY      EQU   £B000
SPRPOS   EQU   £B001
BITPOS   EQU   £B003
SPRTAB   EQU   £B004
DFCC     EQU   £5C84
         ORG   £B100
INIT     LD    HL,(DFCC)
         JR    SAVSCR
MOVSPR   LD    D,0
         CALL  UNDER
         LD    HL,(SPRPOS)
         LD    A,(KEY)
         CP    5
         JR    NZ,L0
         CALL  LMOVE
         JR    SAVSCR
L0       CP    6
         JR    NZ,L1
         CALL  BELOW
         JR    SAVSCR
L1       CP    7
         JR    NZ,L2
         CALL  ABOVE
         JR    SAVSCR
L2       CP    8
         RET   NZ
         CALL  RMOVE
SAVSCR   LD    (SPRPOS),HL
         LD    D,1
         CALL  UNDER
         CALL  PRSPRT
         RET
UNDER    LD    HL,(SPRPOS)
         LD    IX,SPRTAB
         LD    C,8
LINE     LD    B,2
BYTE     BIT   0,D
         JR    NZ,SVESCR
WIPOUT   LD    A,(IX+£10)
         LD    (HL),A
         JR    CONT
SVESCR   LD    A,(HL)
         LD    (IX+£10),A
CONT     INC   HL
         INC   IX
         DJNZ  BYTE
         DEC   C
         RET   Z
         INC   IX
         DEC   HL
         DEC   HL
         PUSH  BC
         PUSH  DE
         CALL  BELOW
         POP   DE
         POP   BC
         JR    LINE
BELOW    LD    A,7
         AND   H
         CP    7
         JR    Z,BDIFCB
BSAMCB   INC   H
         RET
BDIFCB   LD    DE,£20
         ADD   HL,DE
         LD    A,£E0
         AND   L
         RET   Z
BSAMSB   LD    A,H
         SUB   7
         LD    H,A
         RET
ABOVE    LD    A,7
         AND   H
         JR    Z,ADIFCB
ASAMCB   DEC   H
         RET
ADIFCB   LD    DE,£20
         AND   A
         SBC   HL,DE
         LD    A,£E0
         AND   L
         CP    £E0
         RET   Z
ASAMSB   LD    A,H
         ADD   A,7
         LD    H,A
         RET
LMOVE    LD    IX,BITPOS
         LD    A,(IX+0)
         DEC   A
         AND   7
         LD    (IX+0),A
         LD    C,A
         INC   IX
RMOVE    LD    IX,BITPOS
         LD    A,(IX+0)
         INC   A
         AND   7
         LD    (IX+0),A
         LD    C,A
         INC   IX
         LD    B,8
RPAIR    LD    E,(IX+0)
         LD    D,(IX+1)
         RRC   E
         PUSH  AF
         RLC   E
         POP   AF
         RR    D
         RR    E
         LD    A,0
         CP    C
         JR    NZ,RSTORE
         LD    A,D
         LD    D,E
         LD    E,A
RSTORE   LD    (IX+0),E
         LD    (IX+1),D
         INC   IX
         INC   IX
         DJNZ  RPAIR
         LD    A,0
         CP    C
         RET   NZ
         INC   HL
         RET
PRSPRT   LD    HL,(SPRPOS)
         LD    IX,SPRTAB
         LD    C,8
PRLINE   LD    B,2
PRBYTE   LD    A,(IX+0)
         CPL
         LD    D,A
         LD    A,(HL)
         AND   D
         OR    (IX+0)
         LD    (HL),A
         INC   HL
         INC   IX
         DJNZ  PRBYTE
         DEC   C
         RET   Z
         DEC   HL
         DEC   HL
         PUSH  BC
         CALL  BELOW
         POP   BC
         JR    PRLINE
         LD    B,8
LPAIR    LD    E,(IX+0)
         LD    D,(IX+1)
         RLC   E
         PUSH  AF
         RRC   E
         POP   AF
         RL    D
         RL    E
         LD    A,7
         CP    C
         JR    NZ,LSTORE
         LD    A,D
         LD    D,E
         LD    E,A
LSTORE   LD    (IX+0),E
         LD    (IX+1),D
         INC   IX
         INC   IX
         DJNZ  LPAIR
         LD    A,7
         CP    C
         RET   NZ
         DEC   HL
         RET
```

Cómo usar estos programas

1) Digite y guarde (SAVE) "Programa 1"
2) Digite y ejecute "Programa 2", que tomará el código máquina de la memoria y lo guardará (SAVE) en la cinta, a ser posible después de "Programa 1"
3) Cargue (LOAD) el "Programa 1", que se ejecutará automáticamente

En primera línea

Desde sus modestos comienzos, Motorola ha ido creciendo hasta llegar a la posición que ocupa actualmente: uno de los primeros fabricantes de componentes microelectrónicos del mundo



Robert Galvin, presidente de Motorola

Al igual que muchas otras empresas de éxito, Motorola empezó como el negocio de un solo hombre. La fecha de creación de la firma se remonta a 1928, cuando Paul Galvin fundó la Galvin Manufacturing Corporation, en Chicago, que se especializó en la producción de receptores de radio para el hogar. Durante los años treinta la empresa se diversificó, fabricando radios para la policía y para automóviles bajo la marca comercial "Motorola". En los años cuarenta la empresa (cuyo nombre ahora es Motorola Incorporated) fue una de las primeras firmas de electrónica en producir semiconductores.

Paul Galvin falleció en 1959 y lo sucedió como presidente su hijo, Robert. Durante la década siguiente otros fabricantes, en particular japoneses, empezaron a competir con Motorola en los mercados del semiconductor y de la electrónica de consumo. La recesión mundial de mediados de los setenta hizo que la empresa sufriera enormes pérdidas y se viera forzada a replantear su estrategia. Se contrató nuevo personal, gran parte del cual provenía del rival por antonomasia de Motorola, Texas Instruments, y se tomó la decisión de abandonar el campo de la electrónica, en el cual la compañía ya no podía competir, para concentrarse, en cambio, en la microelectrónica de alta tecnología.

Ello implicó la venta de parte del activo de la firma (en particular el negocio de televisores en color), la inversión de fuertes sumas en investigación y desarrollo, y la adquisición de empresas en zonas nuevas en las que Motorola deseaba causar un gran impacto. Ello representaba un riesgo considerable, pero en aquel momento las alternativas que se le presentaban eran escasas.

Las oficinas centrales de Motorola en Illinois (Estados Unidos)



La apuesta parece haberse ganado. Durante la segunda mitad de la década de los setenta Motorola quedó muy a la zaga de las principales firmas fabricantes de semiconductores, pero después de grandes inversiones en nueva tecnología la empresa ahora afirma estar pisándole los talones a Texas Instruments, líder del mercado. Como comenta Robert Galvin: "Las empresas que solían hacerle la competencia a Motorola han quedado en el camino porque no se han adaptado al medio".

Motorola ha seguido teniendo problemas para lograr que sus productos salieran a la venta en el momento adecuado. A mediados de los setenta, cuando la industria del microordenador estaba en pañales, el microprocesador Motorola 6800 fue superado, en cuanto a volumen de ventas, por el Mostek 6502, que fue adoptado por Apple para sus ordenadores personales, de tan fabuloso éxito, y por el Intel 8085 y el Zilog Z80, que utilizan los ordenadores con CP/M. La empresa introdujo el 6809 en 1976; éste fue reconocido a nivel general como el mejor microprocesador de ocho bits de todos los existentes, pero la carrera por el mercado masivo ya se había perdido y el chip sólo apareció en unos pocos micros personales, como el Tandy Color y el Dragon.

No obstante, la empresa ha seguido realizando grandes inversiones en investigación ("para sacar la máxima ventaja lo antes posible", según declara Robert Galvin) y ahora está mucho mejor situada en la carrera por el mercado de 16 bits. El microprocesador 68000 se lanzó en 1979, aunque no estuvo disponible ampliamente hasta 1982. Este procesador es el que ha adoptado Apple para sus microordenadores Lisa y Macintosh, y Sinclair Research para su QL. Se trata de un dispositivo extremadamente potente que contiene 17 registros de 32 bits, un bus de datos de 16 bits y un bus de direcciones de 24 bits.

Motorola continúa desarrollando nuevos productos en sus centros de investigación de Phoenix (Arizona), Ginebra (Suiza) y East Kilbride (Escocia). La fábrica de East Kilbride fabrica chips CMOS (semiconductores de óxido metálico complementario) y MOS (semiconductores de óxido metálico) para una amplia gama de aplicaciones. En la actualidad, la empresa está organizada en cinco grupos, que se ocupan de comunicaciones, semiconductores, sistemas de información, electrónica para la automoción e industrial, y electrónica para la administración. A pesar de la baja rentabilidad de algunos de sus departamentos, Motorola vio ascender sus ventas a 1,26 billones de dólares en el primer trimestre de 1983 y parece dispuesta a mantener la sólida posición que ocupa en estos momentos en el mercado mundial de la microelectrónica.


R T Y U I O
D F G H J K L
X C V B N M

10042
9 788485 822836